НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО _____

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Додаток для моніторингу здоров'я пацієнта»

Виконав (-ла):
студент (-ка) IV курсу, групи ІО-63
Голота Михайло Михайлович _____

Керівник:
Доцент кафедри ОТ, с.н.с., к.т.н.
Антонюк Андрій Іванович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
доц., к.т.н.,
Орлова Марія Миколаївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «6.050102»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО_____
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Голоти Михайла Михайловича

1. Тема проєкту «Додаток для моніторингу здоров'я пацієнта» керівник проєкту Антонюк Андрій Іванович, с. н. с., к. т. н., доцент., затверджені наказом по університету від «07» травня 2020р. №1081-с

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту: технічна документація, теоретичні дані, інтернет-публікації за темою роботи

4. Зміст пояснювальної записки: проведення аналізу предметної області та існуючих передумов для розробки, проєктування і розробка додатка для моніторингу здоров'я пацієнта, провести тестування додатку

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

6. Консультанти розділів проєкту*

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., проф.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019	
2	Вивчення та аналіз завдання	22.12.2019	
3	Проведення аналізу предметної області	23.02.2020	
4	Проведення аналізу вхідних даних та можливих гіпотез до побудови системи	08.03.2020	
5	Проектування та розробка додатку для моніторингу здоров’я пацієнта	13.03.2020	
6	Проведення тестування розробленого додатку	20.03.2020	
7	Оформлення матеріалів роботи	20.05.2020	
8	Передзахист	26.05.2020	
9	Захист	…	

Студент Михайло ГОЛОТА

Керівник Андрій АНТОНЮК

Анотація

В даній роботі був розроблений веб додаток, який дозволяє користувачу слідкувати за станом свого здоров’я не виходячи з дому. В ході дослідження аналогів було виявлено, що жодна з програм не має в собі подібного функціоналу. Під час розробки систему було запроваджено клієнт-серверною архітектурою та застосуванням технології Prograssive Web App, що забезпечило можливість широкого функціоналу, зокрема кроссплатформенності. Також була детально проаналізована область медицини, та її основні терміни в сфері терапії. Як наслідок був розроблений веб-додаток, що включає в себе весь потрібний функціонал та забезпечує доступ до додатку з будь-якої платформи . Призначенням програми є забезпечення зручної форми комунікації лікуючого лікаря з пацієнтом та повний моніторинг його стану здоров’я, крім цього забезпечується документообіг між лікарем та його пацієнтом.

Annotation

In this work, a web application was developed that allows the user to monitor their health without leaving home. During the study of analogues it was found that none of the programs has such functionality. During the development, the system was implemented by client-server architecture and the use of Prograssive Web App technology, which provided the possibility of a wide range of functionality, including cross-platform. The field of medicine and its main terms in the field of therapy were also analyzed in detail. As a result, a web application was developed that includes all the necessary functionality and provides access to the application from any platform. The purpose of the program is to provide a convenient form of communication between the attending physician and the patient and full monitoring of his health, in addition, the flow of documents between the doctor and his patient.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 6309. 00.000 ВП	Відомість дипломного проєкту	1	
3	А4	ДП 6309. 01.000 ТЗ	Технічне завдання	3	
4	А4	ДП 6309. 02.000 ПЗ	Пояснювальна записка	70	
5	А3	ДП 6309. 03.000 Д1	Схема алгоритму аутентифікації користувача.	1	
6	А3	ДП 6309. 04.000 Д2	Сховище даних. Схема даних.	1	
7	А3	ДП 6309. 05.000 Д3	Взаємодія класів в системі. Діаграма класів.	1	

				ДП 6309. 00.000 ВП		
	ПІБ	Підп.	Дата			
Розробн.	Голота М.М.			Відомість дипломного проєкту	Лист	Листів
Керівн.	Антонюк А.І.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. ОТ Гр. ІО-63	
Н/контр.	Сімоненко В.П.					
Зав.каф.	Стіренко С. Г.					

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проєкту
освітньо-кваліфікаційного рівня__бакалавр___

на тему: “ Додаток для моніторингу здоров’я пацієнта”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку додатка для моніторингу здоров’я пацієнта.

Область застосування: моніторинг здоров’я пацієнтів.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить покращення процесу моніторингу здоров’я та покращення комунікації пацієнта з терапевтом.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту є розробка додатка для моніторингу здоров’я пацієнта.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, довідники з програмованих логічних інтегральних схем, публікації в Інтернеті за даним питанням.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розроблюваного продукту

- Розробка серверної частини додатку.
- Проектування баз даних
- Розробка клієнтської частини додатку

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 7, MS Windows 8/8.1, MS Windows 10, GNU/Linux, MacOS
- PHP 5.6 і вище

5.3. Вимоги до апаратного забезпечення

- Процесор з тактовою частотою не нижче 2.2 ГГц
- Оперативної пам’яті не менше 2 Гбайт

Пояснювальна записка
до дипломного проєкту
на тему: «Додаток для моніторингу здоров’я пацієнта»

Київ - 2020 року

ЗМІСТ

ВСТУП

Сьогодні будь-яка галузь прагне розвитку та автоматизації, і медицина, як одна з найнеобхідніших людству не стоїть осторонь. Сучасні ІТ розробки роблять позитивний вплив на розвиток нових способів організації медичної допомоги населенню. Проведення онлайн-консультацій для пацієнтів і персоналу, обмін інформацією про хворих між різними установами, дистанційне фіксування фізіологічних параметрів, контроль за проведенням операцій в реальному часі – все це реалії сьогоднішнього дня. Такі веб додатки виводять інформатизацію охорони здоров'я на новий рівень розвитку, позитивно позначаючись на всіх аспектах його діяльності. Актуальність розробки додатків для медичних центрів, приватних клінік буде зростати з кожним днем. Мобільні додатки планують розпорядок дня, меню правильного харчування, а також режим тренувань, розповідають користувачу про стан здоров'я, і допомагають контролювати прийом ліків. Крім цього, через додаток можна зв’язатись з лікуючим лікарем, який дасть певні рекомендації щодо змін стану здоров’я.

Таким чином, даний дипломний проєкт присвячений розробленню веб додатку, який дозволить реформувати форму комунікації з лікарями та спростить моніторинг стану здоров’я користувача.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

Мета роботи: Створити кроссплатформенний додаток для моніторингу здоров'я пацієнта.

1.1 Огляд проблеми, яка вирішується ПЗ

Веб додаток покликаний допомогти самостійно діагностувати захворювання за симптомами став реалією сьогодення. Розробка веб додатків для медичних центрів допомагають вирішити низку проблем, адже підвищується доступність медичних послуг, люди можуть отримати кваліфіковану допомогу від досвідчених лікарів віддалено. Таким чином відбувається автоматизація бізнес-процесу. Внаслідок аналізу існуючих додатків було виявлено, що вони не в повній мірі можуть вирішувати головну задачу – спрощувати моніторинг стану здоров'я. Основними причинами стали відсутність: можливості повного контролю лікаря за пацієнтом, повного переліку можливих скарг на стан здоров'я, збереження діагностичних обстежень, інших довідок та обмеженість автоматизованої системи переліку діагнозів, яка дозволяє обрати зі схожими симптомами, але не точний. Звісно, існує можливість описати власноруч симптоми та скарги, проте це у свою чергу потребує часу та значно ускладнює використання додатку.

Тому головними проблемами є: ведення обліку пацієнтів клінік, спостереження дистанційно за їх станом, збереження та передача результатів діагностичних обстежень, проведення віддаленого лікування та контроль правильності призначеного лікування, його дієвості, наявність автоматизованої системи повного переліку симптомів, скарг та діагнозів.

Виходячи з цього метою даної роботи є створення програмного рішення, яке дозволить реформувати існуючу комунікацію з лікарями, а також значно спростить моніторинг стану здоров'я.

1.2 Опис вимог до розроблюваного ПЗ

Сформовано ряд вимог, яким має веб додаток для моніторингу здоровя. Перш за все, додаток має бути кроссплатформеним. Оскільки в наш час багато додатків працює лище на платформі iOS чи Android. Другим та не менш важливим критерієм є зрозумілість інтерфейсу. Також потрібно забезпечити коректну роботу додатку при слабому звязку або відсутності мережі Інтернет. Додаток мусить містити 3 інтерфейси: адмін, терапевт та пацієнт. Розглянемо детальніше: інерфейс адміна дає змогу створити терапевта та пацієнта, створити опитувальник та репорт. В інтерфейсі терапевну має бути функціональність для створення облікового запису пацієнта, змога редактування, обмін документами та привязкой опитувальників. Пацієнт має змогу проходити опитування, завантажувати та обмінюватися документами з терапевтом, та запрошувати членів сімї. Додаток мусить містити інтерфейси авторизації, автентифікації та відновлення паролю. Так як продукт націлений не тільки на український маркет, сайт має підтримувати інтернаціоналізацію (українську та англійську мову).

Таким чином, можна узагальнити вищезгадані вимоги:

1. Кроссплатформенність.
2. Робота при відсутності марежі Інтернет.
3. Підтримка інтернаціоналізації.
4. “User friendly” інтерфейс.
5. Обмін документами.

Отже, розглянуто зазначені вимоги, згідно з якими проведено детальний порівняльний аналіз існуючих програмних рішень. Також враховано їх переваги та недоліки при розробленні програмного застосунку.

1.3 Аналіз існуючих рішень

При розгляді питання про створення додатку для моніторингу здоровя пацієнта було встановлено, що найбільш популярним рішенням є використання технологій, що працюють лише на платформі iOS або Android.

Таким чином, проаналізовано програмні засоби, розроблені відомими компаніями.

1.3.1 Аналіз програмного рішення Apple HealthKit

Дане програмне забезбечення покращує медичне обслуговування. Додаток зберігає дані про здоровя та фізичні активності користувача на iPhone та Apple Watch. Користувач може передавати дані своєму лікарю. Реалізована на платформі iOS з використанням мови програмування Swift.

Рис. 1.1. Ілюстрація роботи Apple HealthKit

Незважаючи на те, що даний додаток є досить популярним, в ньому є багато

недоліків. Перш за все це те, що він працює лише на платформі iOS, що позбавляє можливості користуватися ним на другому девайсі. Також додаток немає інтернаціоналізації, тобто ним можуть користуватися лише англомовні юзери. Із врахуванням визначених раніше вимог для даного програмного рішення виділено наступне: зрозумілий інтерфейс, багатофунціональність, підтримка відсутності мережі інтернет, здатність завантажити та обмінятися документам з терапевтом. При цьому слід зазначити, що додаток може працювати лише на платформі iOS та підтримує лише англійську мову.

1.3.2 Аналіз програмного рішення Goggle Fit

Подібно до Apple HealthKit, розширює можливості медичного обслуговування.

Програма збирає найрізноманітнішу інформацію про здоров'я користувача, і він може ці дані надсилати своєму лікарю. Даний додаток має широкий функціонал: проходження опитувальників, завантаження даних та комунікація з терапевтом. Платформи які підтримують даний додаток - iOS та Android. Також важливим фактором є те, що в інтерфейсі є вибір мови користувача, що збільшує частку ринку для цього додатку. З мінусів, які потрібно виокремити - це перш за все великий обєм потрібної памяті, що не дозволяє деяким користувачам встановити даний додаток. Також ссилаючись на дане джерело [1] можна стверджувати, що багато юзерів мають проблеми з оновленням додатку.

Рис. 1.2. Приклад роботи Google Fit

Відповідно до вимог, описаних вище, проаналізовано дане програмне рішення. Щодо переваг, то додаток є багатофункціональним, має інтернаціоналізацію, підтримується на iOS та Android. Слід відмітити наявність інтуїтивно зрозумілого інтерфейсу. Серед недоліків потрібно виділити наступне: відсутність обміну документами, великий обєм потрібної памяті та проблеми з обновленням додатку.

1.3.3 Аналіз програмного рішення Medisafe

Мобільний додаток Medisafe по функціоналу набагато ширше попереднього: тут можна відстежувати будь-які прописані ліки. Причому програма дозволяє додавати як регулярні ліки (курсові), так і разові, які ви взяли по самопочуттю (знеболююче або снодійне, наприклад). Для консультацій зі своїм лікарем ви можете вивантажити звіти у форматі PDF і XLS[3], вносити дані про своє тиску, рівні глюкози в крові, вага і інші параметри. Є і менш значущі, але

зручні опції: синхронізація нагадувань з годинником Android Wear, установка різних мелодій на окремі препарати. На жаль, розробники поки не включили функцію, яка буде показувати початок і кінець прийому ліків, скільки таблеток прийнято і скільки ще залишилося (програма поки враховує тільки число днів курсу), і не має можливості встановити перерву в прийомі медикаментів.

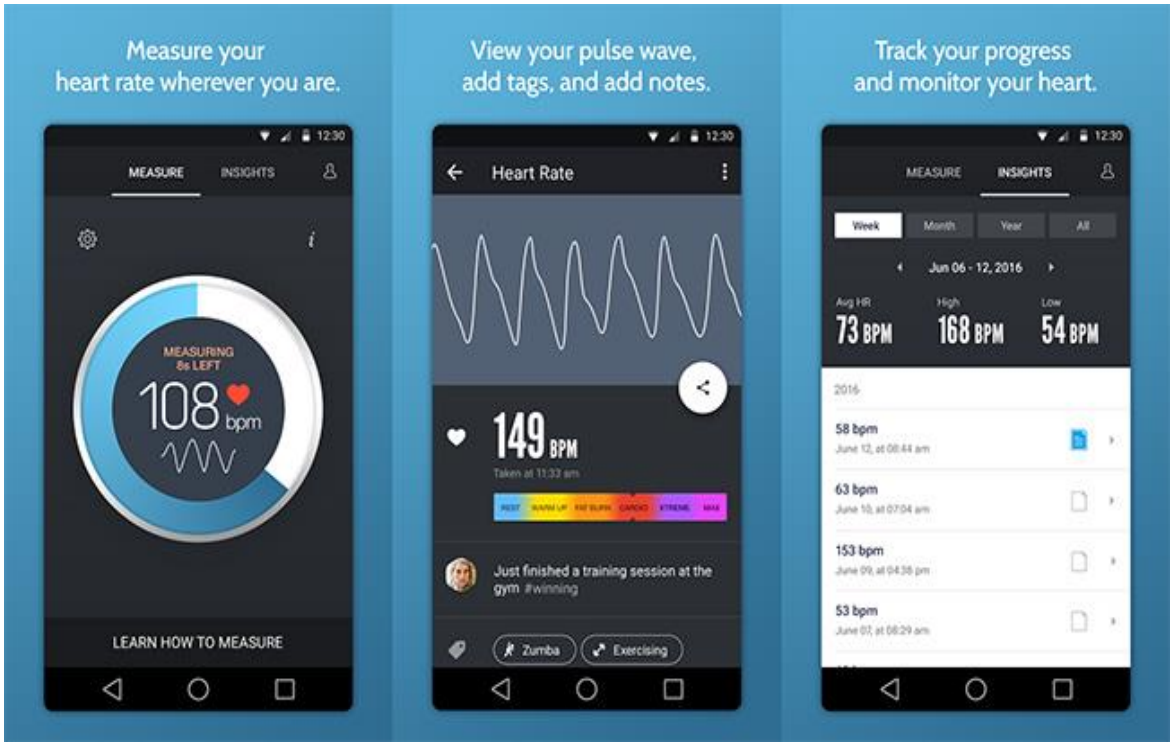


Рис. 1.3. Ілюстрація роботи Medisafe

З точки зору обраних вимог для аналізу програмного рішення, дана система має ряд переваг, а саме: багатофункіональність, робота при відсутності марежі Інтернет, підтримка інтернаціоналізації та інтуїтивно-зрозумілий інтерфейс для взаємодії із системою. Але слід враховувати і те, що даний додаток доступний тільки для платформи Android[4].

ВИСНОВКИ ДО РОЗДІЛУ 1

Виходячи з отриманих даних після проведення аналізу, можна стверджувати, що згадані системи повністю не вирішують поставлені задачі. Більшість із описаних програм орієнтовані на використання лише на телефонах, деякі лише в браузері, в деяких не вистачає функціоналу, або вони мають проблеми з оновленням.

Отже, сформовано порівняльну таблицю результатів аналізу програмних рішень (див. табл. 1.1).

Таким чином, для програмного рішення задачі моніторингу здоровя пацієнта виділено ряд вимог:

6. Система має працювати кроссплатформенно.
7. Підтримувати інтернаціоналізацію.
8. Працездатність під час відсутності інтернету.
9. Займати небагато памяті.
10. Простота оновлення версії
11. Інтуїтивно зрозумілий інтерфейс

Таблиця 1.1 Результати аналізу існуючих програмних рішень

Критерії / Програмні рішення	Кроссплатформенність	Підтримка інтернаціоналізації	Інтуїтивно зрозумілий інтерфейс	Підтримка відсутності інтернету
Apple HealthKit	-	+	+	+
Goggle Fit	-	+	+	+
Medisafe	-	+	+	-

Також програмне рішення має вирішувати наступні проблеми:

1. Кроссплатформеність.
2. Проблему моніторингу здоровя пацієнта.
3. Комунікація пацієнта та терапевта.
4. Додаток займає багато пам'яті пристрою.

Отже, як результат після проведення аналізу існуючих рішень відповідно до вимог маємо наступні завдання для створюваного програмного застосунку:

1. Розроблення клієнтської частини додатку:
    1.1. Створення трьох інтерфейсів з повним фунціоналом.
    1.2. Забезпечення кроссплатформенності.
    1.3. Підтримка роботи офлайн.
2. Створення backend частини додатку.
3. Реалізація інтернаціоналізації.
4. Реалізація інтуїтивно-зрозумілого інтерфейсу користувача.

РОЗДІЛ 2.
ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір технології для реалізації кроссплатформенності

Відповідно до поставленої задачі, а саме створення додатку який можна буде використовувати на будь якому пристрої, випливає необхідність вибору оптимальних засобів для реалізації кроссплатформенності[5].

PWA або прогресивне веб-додаток (англ. Progressive web app) - гібрид сайту і додатки для мобільних пристроїв.

Це технологія, яка додає на сайт функціональність програми. У десктопном браузері progressive web app залишається звичайним сайтом[6]. А коли відвідувач відкриває його в мобільному браузері, PWA перетворюється на гібрид сайту і додатки.

Після установки PWA створює кеш сайту. Це вирішує два завдання: підвищує швидкість завантаження і робить сайт доступним в офлайн режимі[7]. Тобто завдяки технології PWA сайтом можна користуватися навіть без підключення до інтернету.

Впровадження PWA дає суттєві результати. Ось кілька прикладів

- Tinder завдяки технології скоротив час завантаження сторінок з 11,9 до 4,69 секунди. PWA Tinder на 90% «легше», ніж їх нативний додаток.

- PWA Uber майже нічого не важить і вантажиться за 3 секунди навіть в мережах 2G.

- OLX завдяки PWA підвищив CTR оголошень на 146% і зменшив кількість відмов на 80%.

У браузерах Google, Opera, Firefox і Microsoft PWA можна завантажити на будь-яких гаджетах, незалежно від розміру екрану і інших специфікацій. Крім того, розробники даних браузерів будуть пропонувати користувачам встановити PWA при другому відвідуванні сайту[8].

- Не потрібно API з підтримкою зворотної сумісності. У випадку з PWA користувачі запускають ту ж версію коду сайту (на відміну від класичних

додатків).

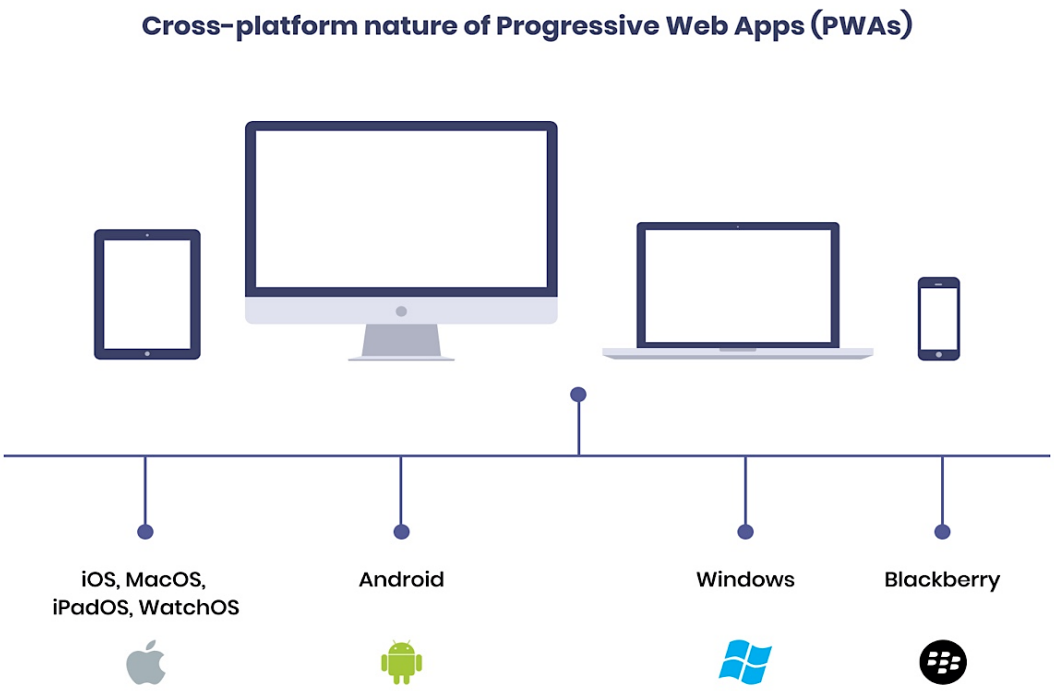


Рис. 2.1. Концепція PWA

Надійність (Reliable) - додаток завантажується і показується відразу ж, незалежно від статусу і якості мережевого з'єднання.

Швидкість (Fast) - взаємообмін даними по мережі відбувається швидко.

Привабливість (Engaging) - робить для користувача досвід роботи з додатком комфортним і приємним.

З точки зору Google, саме це відокремлює зараз за зовнішнім виглядом і відчуттями (look and feel) веб-сайти від нативних додатків[9].

PWA і нативні додатки

Те, що PWA зовні схожі на нативні додатки, є, скоріше, косметичним рішенням (хоча і важливим для користувача з психологічної точки зору). А ось те, що вони схожі внутрішньо (всі основні ресурси програми можна зберігати на клієнті, по мережі буде передаватися тільки мінливий контент) - це величезне досягнення.

Можна навіть назвати це прихованою революцією. По суті, браузер

використовується як якась віртуальна машина, яка зберігає і запускає в собі PWA додаток[10]. Як Андроїд є віртуальною машиною для андроїд-додатків, так і браузер стає віртуальною машиною для PWA. [11]Як нативний додаток звертається через файлову систему до своїх ресурсів, так само і PWA звертається до своїх ресурсів - нехай по HTTP, але зберігаються локально. І все це однаково працює на всіх основних браузерах і на всіх основних платформах.

2.1.1 Service Worker

Це проксі-шар між фронтендом і бекендом, що знаходиться в браузері. Всі запити браузера йдуть через нього. Цей поділ на дві незалежні частини дозволило зробити перехід звичайного веб сайту в PWA максимально простим.

Зі сховищ у Service Worker'а є доступ до Cache Storage для web ресурсів, і IndexDB для даних. Але, найголовніше, повна свобода для реалізації бізнес логіки.

Можна, наприклад, прийняти запит від браузера, перевірити стан мережі, взяти дані зі сховища, зробити з ними операції та повернути якийсь результат назад в браузер - який буде “думати”, що відповідь йому прийшла від сервера. Два браузерних шари (клієнтський фронтенд і Service Worker) дозволяють писати повноцінні програми.

PWA не залежить від фреймворків, це чистий JavaScript, хоча навіть фахівці Google на Хабре чогось радять використовувати Бібліотечні генератори коду. Service Worker прекрасно пишеться руками, і це потрібно, щоб добре розуміти і контролювати логіку роботи твого додатки.

З програмістської точки зору Service Worker є JavaScript файл підключається в html коді сторінки. У ньому розробник визначає логіку роботи з надходженням з фронтенду запитів і іншу функціональність.

2.1.2 HTTPS

PWA вимагає, щоб всі ресурси сайту передавалися по HTTPS протоколу. SSL сертифікат можна отримати безкоштовно. Але критично, щоб на сайті не було посилань на незахищений ресурси - деякі браузери просто не будуть відображати сайт в цьому випадку.

Основна проблема яка зустрічається в таких це картинки[12]. Часто редактори або коментатори ставлять в матеріал посилання на картинки з інтернеті. Необхідно картинки зберінати до себе, або на сервіс з доступом по HTTPS.

2.1.3 Application Shell

App shell - це просто скелет графічного інтерфейсу, шаблон. Для прикладу, візьмемо середній сайт с Header, двома колонками і іншим[13]. Грубо кажучи, виріжемо з нього контент поточної сторінки і всю динамічну інформацію, що залишилася статика - app shell.

Суть в тому, що app shell зберігається на клієнті і завантажується при запуску програми, а потім вже в нього вантажиться з мережі динамічна інформація. І поки вона вантажиться, app shell повинен не відображати помилку, а наприклад показати «лоадери»[14].

2.1.4 Web App manifest

JSON файл, декларативно визначає для браузера назву програми, іконку, як буде виглядати PWA (fullscreen, standalone і ін.) І деякі інші параметри[15]. Дозволяє «встановити» PWA як окремий додаток на домашній екран смартфона.

2.1.5 Push Notifications

Якщо посерфити інтернет з Chrome DevTools, відкритим на вкладці Application, то можна побачити, як мало сайтів використовують PWA технології.

А 90% тих, що використовують, роблять це тільки заради Push Notifications.

Поки що це найпопулярніша і сама зловживана технологія PWA[16].

2. 2 Вибір технологій для реалізації backend частини додатку

Дивлячись на основні правила розробки веб додатку сформуєму наступну архітектуру автоматизованої системи[17]. Система буде розрозроблена за допомогою використання програмного коду PHP. PHP - це широко використовується мова сценаріїв загального призначення з відкритим вихідним кодом. Говорячи простіше, PHP - це мова програмування, спеціально Розроблений для написання web-додатків (сценаріїв), виконуються на web-сервері. Абревіатура PHP означає «Hypertext Preprocessor (Препроцесор гіпертексту)»[18]. PHP досить простий для вивчення. Перевагою PHP є надання web-розробникам можливості швидкого створення динамічно генеруються web-сторінок. Важливою перевагою мови PHP перед такими мовами, як Perl і C полягає в можливості створення HTML-документів з внедренними командами PHP[19]. Значною відмінністю PHP від будь-якого коду, що виконується на стороні клієнта, наприклад, JavaScript, є те, що PHP-скрипти виконуються на стороні сервера[20]. Практичний характер PHP обумовлений п'ятьма важливими характеристиками:

- традиційністю;
- простотою;
- ефективністю;
- безпекою;
- гнучкістю.

PHP надає в розпорядження розробників і адміністраторівгнучкі і ефективні засоби безпеки, які умовно поділяються на дві категорії: засоби системного рівня і засоби рівня додатки[21]. Через те що PHP є вбудовуваним (embedded) мовою, він відрізняється Винятковою гнучкістю по відношенні до потреб розробника. Хоча PHP зазвичай рекомендується використовувати в поєднанні з HTML, він з таким же успіхом інтегрується і в JavaScript, WML, XML і інші мови[22]. Крім того,

добре структуровані додатки PHP легко розширюються в міру потреби (втім, це відноситься до всіх основних мов програмування). Немає проблем і з залежністю від браузерів, оскільки перед відправкою клієнту сценарії PHP повністю компілюються на стороні сервера. В суті, сценарії PHP можуть передаватися будь-яких пристроїв з браузерами, включаючи стільникові телефони, електронні записники, пейджери і портативні комп'ютери, не говорить вже про традиційні PC[23]. Програмісти, що займаються допоміжним утиліт, можуть запускати PHP в режимі командного рядка. За допомогою PHP відбувається зв'язок основних блоків сайту з базою даних, її підключення до сайту. Так само відбувається сортування товарів за різною класифікацією[24].

PHP можна використовувати під управлінням різних операційних систем. Код PHP можна створювати в середовищі безкоштовних Unix подобних операційних систем (Linux, FreeBSD), комерційних версій Unix типу Solaris і IRIX і різних версій Microsoft Windows. Як правило, програми будуть працювати без будь-яких змін в різних середовищах з встановленим PHP.

Користувач має доступ до вихідного коду PHP[25]. На відміну від омерційних закритих програмних продуктів, якщо потрібно щось змінити або додати в цій мові, то це завжди можна зробити.

В якості бази даних ми будемо використовувати MySql. MySQL - це реляційна СУБД, яка представляє собою структуровану сукупність даних, дані зберігаються в ОКРЕМИХ таблицях, мають зв'язку між собою, завдяки чому забезпечується можливість об'єднати при виконанні запиту дані з декількох таблиць. MySql вільна та швидка реляційна база даних з відкритим кодом була створена як альтернатива комерційним системам. Дана база даних найкраще підходить до веб-додатків[26]. Адже швидківсть обробки даних в о обсязі до 500000 записів є найкращою. В неї відкрита, вільна та безкоштовна ліцензія, а також найбільше підходить до більшості хостингових компаній в Україні.

2.3 Вибір технологій для реалізації frontend частини додатку

Як відомо, складовими усіх веб-сайтів є три компоненти, а саме: база даних, серверна та клієнтська частина[27]. Останній з них – просто браузер, який використовується людиною для перегляду сайту. Клієнт може знаходитися у будь-якому місці, де користувач може переглядати сайт: мобільні пристрої, настільні комп'ютери або ноутбуки. У свою чергу, сервер може знаходитися у віддаленому місці у будь-якій точці світу, що зберігає дані про місцезнаходження, при цьому керує внутрішньою архітектурою сайту, обробляє запити та відправляє сторінки на браузер[28]. Таким чином, серверний скрипт виконується веб-сервером, а клієнтський сценарій запускається браузером.

Програмні інструкції на стороні клієнта вбудовані в код розмітки HTML на веб-сайті, який розміщується на сервері мовою сумісним із браузером або скомпільований для зв'язку з ним. Це і є клієнтською мовою програмування.

Відомо, що виконання інструкцій на клієнті завжди розвивається. Воно стає більш простим, швидшим та зручнішим у використанні. У результаті сайти також стають швидшими, ефективнішими за критерієм відношення об'єму виконаної роботи до затраченого часу та при цьому на сервері зменшується кількість роботи, яку необхідно виконати[28].

Розроблення бекенда в проєктах веб-застосунків в основному полягає у написанні компонентів, які користувачі не бачать (підключення до бази даних, системи кешу, продуктивність, безпека та транзакції, тощо). Але для взаємодії з ними необхідно створювати відповідний інтерфейс.

Програмні інструкції виконання впроваджуються всередині коду сторінки і взаємодіють з HTML сайту, вибираючи його елементи, а потім маніпулюють цими елементами, щоб забезпечити інтерактивний процес[29]. Скрипти взаємодіють з каскадним файлом таблиці стилів (CSS), який стилізує відображення сторінки. Він диктує, яку роботу повинен виконати

серверний код, і повертає дані, які повертаються з бекенд у вигляді, що читається браузером[30].

Таким чином сформовано широке уявлення про те, що таке інтерфейсна технологія та розглянуто деякі з найбільш широко використовуваних мов сценаріїв і інтерфейсних фреймворків для реалізації клієнтської частини дипломного проєкту.

Мови майже завжди використовуються в контексті їх фреймворків, які швидко виконують складний код з бібліотеками спільно використовуваного коду і великої кількості надбудов. Розробник може використовувати одну або їх комбінацію при створенні фронтенду сайту[31].

JavaScript - це динамічна мова зі слабо типізованими, динамічно розширюваними об'єктами, які неформально оголошуються в міру необхідності. Функції в ньому є повноцінними об'єктами і зазвичай використовуються у вигляді анонімних замикань. Це робить JavaScript потужнішим мовою, в порівнянні з деякими іншими, часто вживаними для розробки веб-додатків. Теоретично наявність подібних можливостей повинно підвищувати продуктивність програмістів. Але скажімо, що суперечки між прихильниками динамічних і статичних мов, а так же суворої і слабкою типізації досі не вщухли, і навряд чи вщухнуть в найближчому часі[32].

Один з основних недоліків JavaScript - глобальна об'єкт. Це має на увазі, що всі змінні верхнього рівня «звалюються» в глобальний об'єкт, і при використанні одночасно декількох модулів це може призвести до неконтрольованого хаосу. Оскільки веб-додаток зазвичай складається з безлічі об'єктів, можливо, створилися різними службами, то може виникнути ситуація, конфлікту глобальних об'єктів між собою. Але із застосуванням на стороні бекенд (сервера), nodeJS використовує систему організації модулів CommonJS, це означає, що локальні змінні модулів так і будуть локальними, хоча і виглядають як глобальні.

Унікальність JavaScript обумовлена трьома основними аспектами :

1. Повна інтеграція з HTML і CSS.

2. Проста робота з усіма функціоналом.

3. Підтримка усіма браузерами.

JavaScript є імплементацією стандарту ECMAScript[33]. ECMAScrip -

це вбудований розширюваний не має засобів введення-виведення мову програмування, який використовується в якості основи для побудови один скриптових мов. Браузерами зараз підтримується в основному ECMAScript5. Але нас нічого НЕ зупиняє від використання більш сучасного стандарту ECMAScript6, навіть якщо якісь можливості ще НЕ підтримуються браузером[34]. Для цих цілей використовується широко відомий транспайлер Babel.

Транспайлер Babel.js переписує код на ES-2015 (ECMAScript6) в код на попередній стандарті ES5, Який підтримується більш стабільно. Babel.js

можна розділити на дві частини:

1. Транспайлер, Який переписує код.

2. Поліфілл, Який додає методи Array.from,

String, prototype.repeat і інші.

Babel.js дуже легко інтегрується до проєкту через будь-яку систему збирання, яку ви використовуєте. У нашому випадку це webpack.

NPM - (node package manager / пакетний менеджер node.js) є засобом управління залежностями в проєкті Node.js, а так само надає можливість виконувати ряд інших корисних функцій[35]. Потрібно додати, що скрипти npm НЕ заміна збирача, а скоріше його ідеальне доповнення. Хоча можливо за допомогою npm скриптів можна зібрати проєкт, запустить той чи інший пакет, з'єднати кілька файлів в один, перенести файли з однієї папки в іншу. Досить складно це буде реалізовувати тільки npm скриптами, віддамо більшу частину цієї роботи webpack, який створений спеціально з метою збирати проєкт. NPM скрипти відмінно підійдуть для запуску сервера відразу після того як встановляться всі потрібні пакети в проєкт, або для запуску перевірки на стиль коду або помилки за допомогою ESLint або StyleLint.

Всі скрипти описуються до файлі конфігурації packeges.json, в якому по мимо скриптів зберігається опис нашого проєкту, його назву, адресу сховища, контекстна залежностей і інформація про автора. Самі скрипти зберігаються в

об'єкті під назвою «scripts» [15]. Для запуску скрипта потрібно написати в консолі команду npm run і назва самого скрипта. Також є скрипти, які запускаються після певної дії самостійно.

Наприклад, postinstal, Який викликається після установки всіх пакетів і ряд друге команд.

2.3.1 Бібліотека React

React.js володіє простим і зрозумілим API. Для роботи з React потрібно розуміти деяку кількість термінів і відмінності між ними. Елементами називають об'єкти JavaScript, що представляють собою HTML - елементи. наприклад, h1, section або p. Компонентами ж називають елементи React.js, які створюються розробником і можуть носить довільну назву. Як правило вони містять свою специфічну структуру і виконують ряд функцій. елементи і компоненти React створюються за допомогою JSX[36].

React створює з компонентів аналог реального DOM дерева – VirtualDOM і представляє його в браузері. Бібліотека стежить за змінами в віртуально дереві і при зміні його оновлює реальний DOM, щоб реальне і віртуальне дерево були однаковими.

Вище згадані основні концепції, які потрібно знати, щоб починати роботу з React.js. Так само, варто додати в існуванні стану в React, на на практиці ми будемо реалізовувати загальний стан додатки за допомогою Redux. React + Redux досить успішний союз, Який зустрічається досить часто. Компоненти володіють життєвим циклом, наприклад: створення, час життя і демонтаж[37]. Бібліотека надає можливість визначити різні моменти життєвих циклів компонентів і взаємодіяти з ними. При першому використанні компонента, викликають методи життєвого циклам в такому порядку:

- getDafaulProps
- getInitialState
- componentWillMount

• render

• componentDidMount

При подальшому використанні компонента метод getDefaultProps більше не буде викликатися. Компоненти володіють життєвим циклом, наприклад: створення, час життя і демонтаж. Бібліотека дозволяє визначити різні моменти життєвих циклів компонентів і взаємодіяти з ними.

2.3.2 Фреймворк Redux

Даний фреймворк можна використовувати Не тільки з React. Так само він може вживатися разом з Angular, Ember, jQuery і навіть з простим JavaScript. Redux дозволяє описати UI як функцію стану програми і у відповідь на подію дію (action) міняти стан додатки[37].

Ідея дуже проста, весь додаток відповідає єдиному станом або state. У відповідь на ту чи іншу дію користувача або додатки redux викликає свій метод dispatch і передає йому той чи інший action, точніше буде сказати action creator. Action creator - функція, яка створює подія, Повертаючи об'єкт в поле type, якого зазначено назву події. Після виклику методу disaptch відбувається найцікавіше. Фрейморк дає додатком понять, но дещо подія тільки що сталося, і на кожну подію виконується певна логіка дій, яку ми описали заздалегідь в reducer. Reducer – термін в рамках Redux, що позначає обробник подій (actions). Reducer після виконання всіх своїх дій повертає новий стан додатки. Важливо зауваження - стан додатки (state)завжди повертається нове, а не модифікується. Відсутність мутації стану програми гарантує нам відсутність розбіжності даних між моделлю і візуалізації. Після того, як ми отримали новий стан додатки, відбувається звірювання, де саме дані змінили і де потрібно їх замінити[38]. Як наслідок, в DOM замінюються тільки те компоненти, дані яких були змінені в процесі.

ВИСНОВОК ДО РОЗДІЛУ 2

Таким чином, після аналізу описаних вище засобів реалізації додатку зроблено висновки щодо використання їх у розробленні програмного застосунку: кожна із зазначених мов програмування є простою у використанні, не потребує багато часу для конфігурації та дозволяє вирішити всі поставлені завдання для розробки додатку для моніторинга здоров’я пацієнта.

РОЗДІЛ 3.
МОДЕЛЮВАННЯ

3.1 Огляд призначення розроблюваної системи

В даній роботі буде розроблено кроссплатформенний додаток для моніторингу здоров’я пацієнта. Даний додаток повинен надавати користувацький інтерфейс для терапевта, адміна та пацієнта, REST API кінцеві точки та працювати на платформі iOS та Android.

Користувачам буде надана можливість реєстрації та подальшої авторизації. Авторизовані пацієнти можуть проходити опитувальники, завантажувати та обмінюватися документами з терапевтом. Терепевти в свою чергу можуть створювати опитувальники, створювати шаблони, по яким будуть опрацьовуватися дані, обмінюватися документами з пацієнтом.

Цей додаток складатиметься з наступних частин та зв’язків між ними, що зображені на рисунку 3.1.

Як вимоги до основного функціоналу клієнта можна виділити наступні пункти:

1) перегляд списку пацієнтів;
2) форма створення опитуальника
3) привязка опитувальника до пацієнта
4) завантаження документів
5) сортування пацієнтів за певними критеріями;
6) форма реєстрації користувачів;
7) форма логіну;

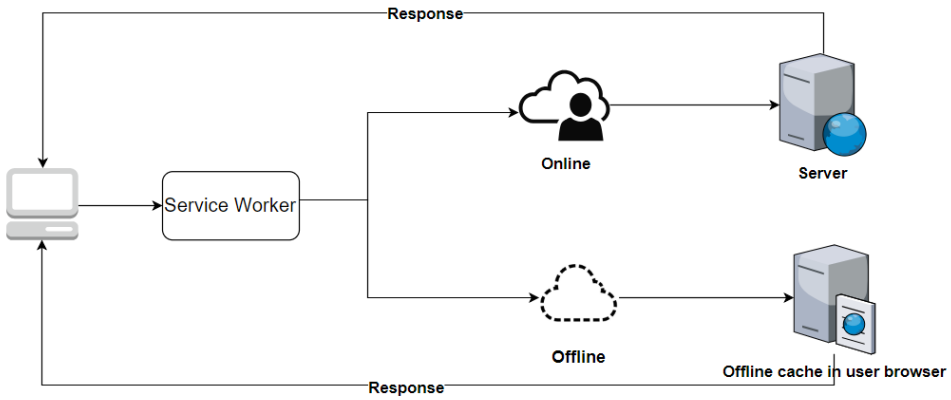


Рис 3.1. Взаємодія елементів верхнього рівня системи

3.2 PWA для вирішення проблеми кроссплатформенності

PWA (Progressive Web Applications; Прогресивні веб-додатки) - це веб-ресурси, які поєднують у собі властивості сайту і мобільного застосування. Зберігаючи зручність використання традиційних веб-сайтів, вони набувають також автономність класичних додатків: з PWA можна працювати як в онлайн, так і в офлайн-режимі.

Розглянемо плюси PWA:

Можливість безбраузерной роботи. Ця функція дає користувачеві доступ до інформації в один клік, а значить, збільшується залученість: клієнт може швидко відвідати ресурс в будь-який зручний момент.

Кросплатформеність[39]. Стандартне мобільний додаток працює тільки на тих платформах, для яких його разрабативали (Android, IOS, Windows). У той же час, PWA здатні працювати майже на всіх апаратних платформах, що Знижує вартість розробки: не потрібно створювати окремий продукт для кожної операційної системи, досить розробити одно універсальне PWA-додаток.

Офлайн-режим. Без Інтернет-з'єднання користувачів може Отримувати інформацію і навіть відправляти HTML-форми[39]. При цьому, дані, які ВВЕЛ відвідувач сайту в форму, тимчасово зберігаються на пристрої, а як

тільки з'явиться зв'язок з інтернетом, вони відправляться на сервер. Уявіть, наприклад, що ви почали заповнювати форму замовлення, а потім зайшли в метро і з'єднання перервалося. На звичайному сайті, якщо ви після цього натисніть кнопку відправки, ваші дані зникнуть; PWA при цьому збереже всю інформацію і відправить її при першій нагоді вже без вашої участі.

Push-повідомлення. Користувачі зможуть отримувати зручні повідомлення про акції, нові статтях і інших подіях вашого ресурсу. Відзначимо, що сприймати ці повідомлення можна навіть без установки PWA (приклад роботи push-повідомлень – дивіться в самому низу сторінки).

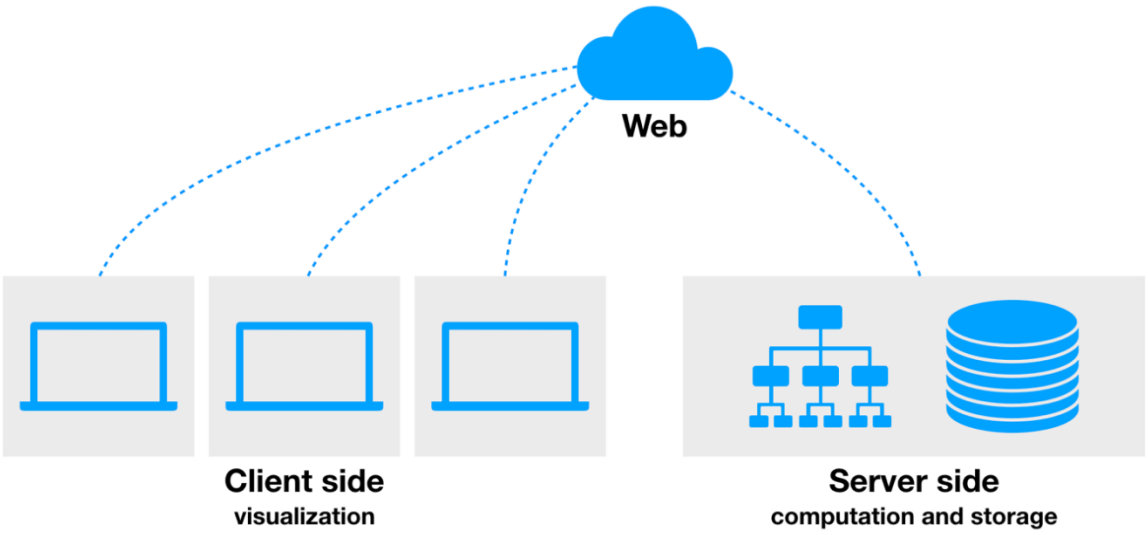


Рис. 3.1. Загальна структура додатку

Сервіс-Воркер, обслуговуючий весь сайт, повинен розташовуватися в корені. Наприклад, мати адресу /service-worker.js. У нашому випадку потрібно саме такий. Якщо ви будете віддавати файл сервіс-Воркер з директорії / js /, наприклад /js/service-worker.js, то він зможе Обробляти тільки те мережеві запити, які починаються з / js / ...

Код ініціалізації сервіс-Воркер в нашому прикладі повинен містити повний список ресурсів, необхідних для коректної відтворення майбутньої сторінки / offline /, всі стилі, зображення і т. П Ми їх попередньо закешіруем за подією install, першому з ланцюжка подій життєвого циклу.

Наступна подія activate. Воно нам стане в нагоді для того, щоб

очистити старий кеш і записи в базі даних. У нашому прикладі для роботи з IndexedDB використовується простий хелпери idb-keyval. Він і його більш прокачаний брат idb є зручними обгортки, промісіфіцірующіми роботу із застарілим морально API IndexedDB.

Що таке «стратегії кешування» і навіщо вони потрібні?

Ресурси, які ми завантажує, грають на сторінці різну роль. Це може бути зображення з логотипом або якась загальна JS-бібліотека, які скоріше за все ніколи НЕ зміняться. Це може бути JSON з коментарями, які оновлюються кожні п'ять хвилин.

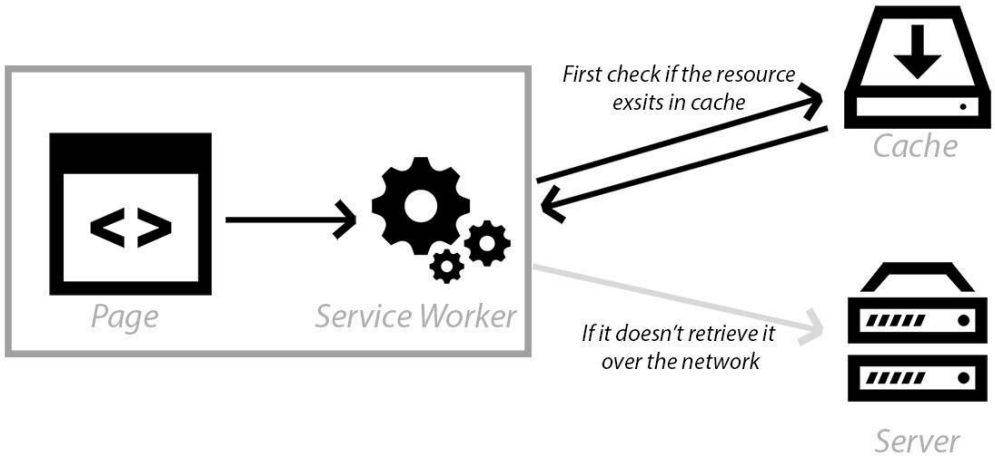


Рис. 3.3.1. Схема кешування

Маніфест визначає наш сайт як додаток для браузера. Вокер- це частина, що залишилася пазла, яка буде забезпечувати логіку роботи програми в тому числі в оффлайн режимі.

Сервіс вокер повністю керований подіями. Це означає що ніякий з ділянок коду НЕ запуститься поки не відбудеться подія, до нього підв'язана. Основні питання, що цікавлять нас - це install і fetch. Ми можемо відстежувати їх додавши наступні терміни коду в файл sw.js

Вокер написаний на чистому JS і служить сполучною ланкою між нашим додатком та мережею. З Вокером (і деякий підтримкою API як Cahe API) нам стає доступним контролю над поведінкою додатки при будь-яких ситуаціях з мережею.

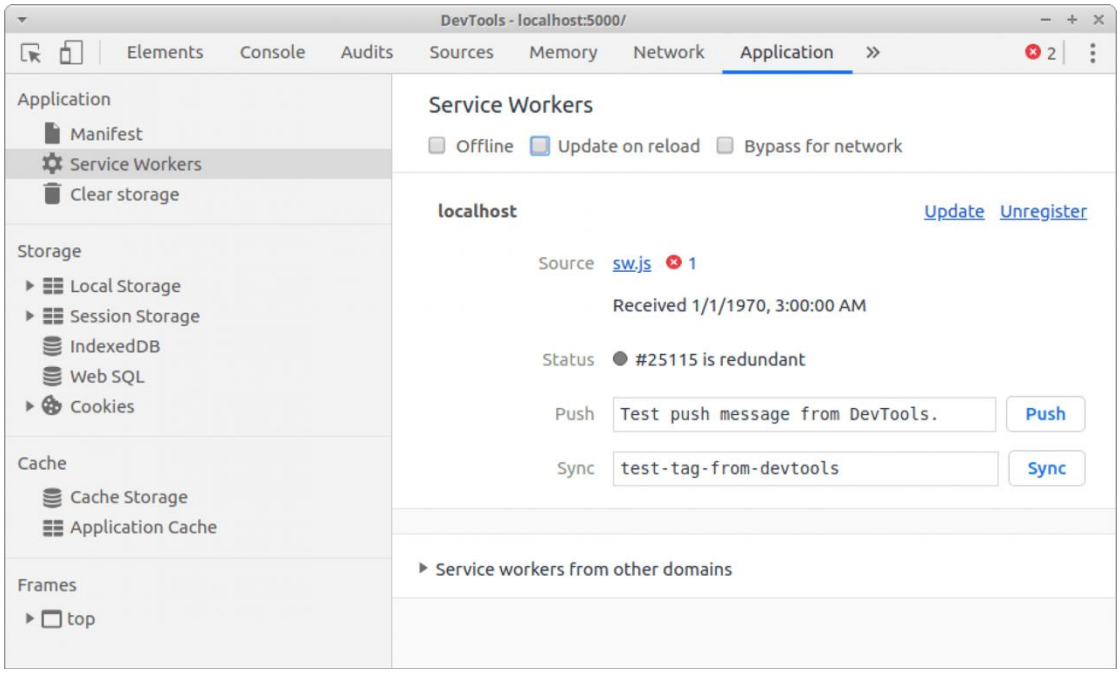


Рис. 3.3.2. Service Workers

3.3 Клієнтська частина

Як ми визначили в розділі 2, доцільно використати бібліотеку Redux та React для створення клієнської частини.

Для використання React досить підключити його пакет в модуль, в якому створюється компонент[40]. У додатку використовуються життєві цикли додатки, оголошені в класі контейнер.

```
componentDidMount () {
this._notificationSystem = this.refs.notificationSystem;
}
```

Метод класу контейнера componentDidMount виконується в рендрінга компонента.

```
shouldComponentUpdate (nextProps) {
if (nextProps.user.authAnswer) {
this._notificationSystem.addNotification ({
message: "Авторизація пройшла успішно!",
level: "success"
```

```
});
setTimeout (() => browserHistory.push ( '/ home'), 1200);
} Else {
this._notificationSystem.addNotification ({
message: `Авторизація не пройдена, спробуйте знову!`,
level: "error"
});
}
return nextProps.user.authAnswer? true: false;
```

}Метод класу контейнера shouldComponentUpdate визначає чи потрібно Рендри заново компоненти, Повертаючи true і false. Метод виконується в момент отримання нових props-ів, або по-іншому властивостей клас компонента.

Таблиця 3.3.1 – Перелік діаграм

№	Тип діаграми	Роль у процесі проєктування
1	Діаграма послідовності	У першому наближенні визначає набір класів та їхню послідовність взаємодії для вирішення певної задачі
2	Діаграма класів	Визначає набір класів та їхню взаємодію задля вирішення певної задачі (виконання процесу)

Діаграма послідовності для роботи додатку наведена у частині графічного матеріалу.

На діаграмі задіяні наступні класи (Таблиця 3.3.2).

Клас	Відповідальність
TherapistContainer ::React.Component	Відображення форм для заповнення даних
PatientContainer ::React.Component	Обгортка для відображення компонент
Request module	Модуль відправлення запитів
Backend API	Модуль виконання запитів
UsersReducer::ReduxStore	Сховище даних

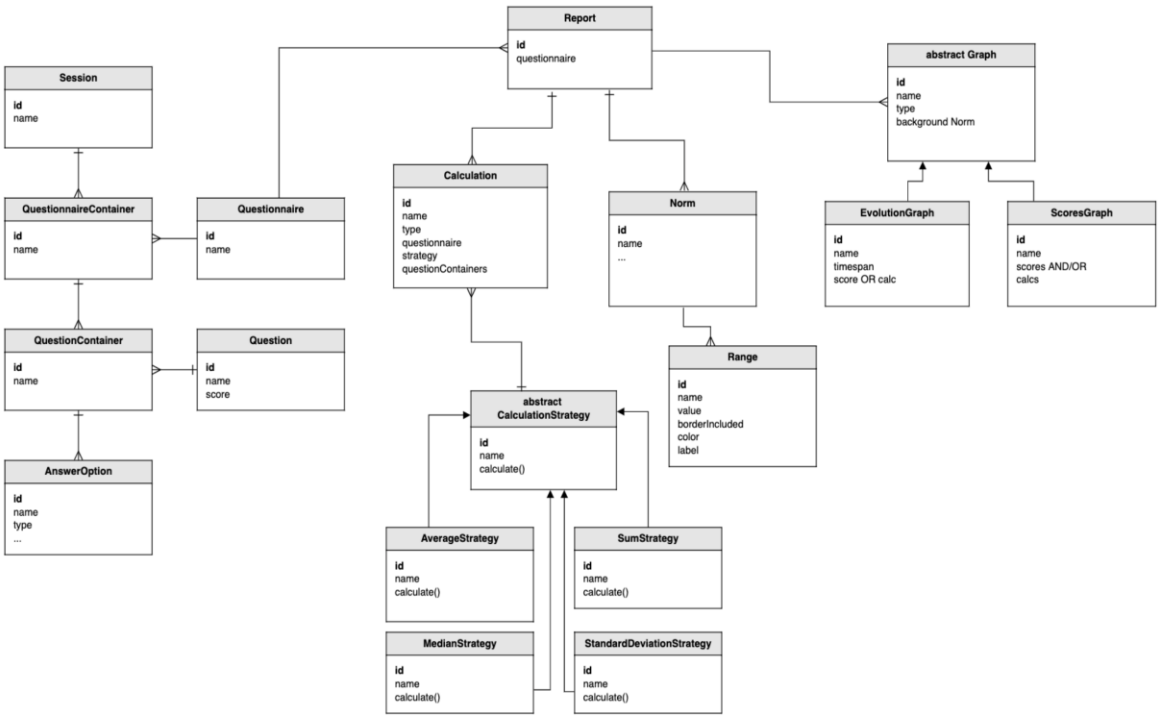


Рисунок 3.1 - ER-діаграма

3.4 Розробка backend частини

На основі отриманих результатів в попередніх розділах, був створений прототип автоматизованого рішення, розроблений з допомогою мова програмування PHP і бази даних для зберігання інформації, створеної на мові MySQL.

Для даного додатка будемо використовувати архітектурний шаблон проєктування MVC. Шаблон MVC описує простий спосіб побудови структури додатка, метою якого є відділення бізнес-логіки від призначеного для користувача інтерфейсу[40]. В результаті, додаток легше масштабується, тестується, супроводжується і звичайно ж реалізується.

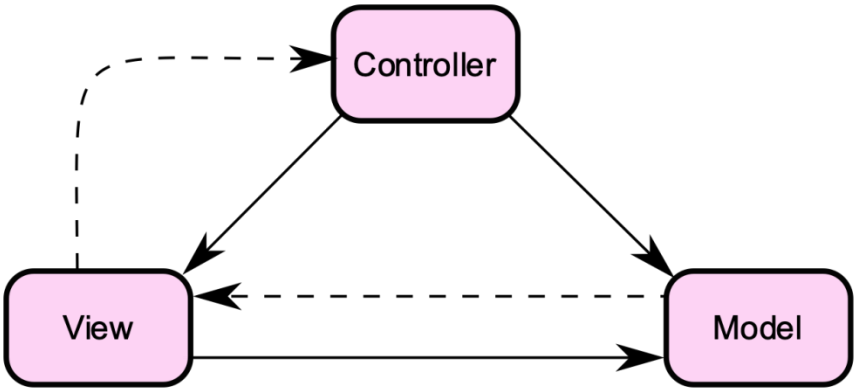


Рисунок 3.4 Схема MVC

Модель - містить бізнес-логіку додатка і включає методи вибірки (це можуть бути методи ORM), обробки (наприклад, правила валідації) і надання конкретних даних, що часто робить її дуже товстою, що цілком нормально[41].

Модель не повинна безпосередньо взаємодіяти з користувачем. Всі змінні, що відносяться до запиту користувача повинні оброблятися в контролері.

Вид - використовується для завдання зовнішнього відображення даних, отриманих з контролера і моделі.

Види містять HTML-розмітку і невеликі вставки PHP-коду для обходу, форматування і відображення даних.

НЕ Повинні безпосередньо звертатися до бази даних. Цим повинні

займатися моделі.

НЕ повинні працювати з даними, отриманими із запиту користувача. Це завдання має виконувати контролер.

Може безпосередньо звертатися до властивостей і методів контролера або моделей, для отримання готових до висновку даних.

Види зазвичай поділяють на загальний шаблон, що містить розмітку, загальну для всіх сторінок (наприклад, шапку і підвал) і частини шаблону, які вживають для відображення даних, що виводяться з моделі або відображення форм введення даних.

Контролер - сполучна ланка, що з'єднує моделі, види і інші компоненти в робоче додаток. Контролер відповідає за обробку запитів користувача. Контролер НЕ повинен містити SQL-запитів. Їх краще тримати в моделях. Контролер НЕ повинен містити HTML і інший розмітки. Її варто виносити в види. У добре спроєктованому MVC-додатку контролери зазвичай дуже тонкі і містять тільки кілька десятків термін коду. Чого не скажеш про Stupid Fat Controllers (SFC) в CMS Joomla. Логіка контролера досить типова і велика, її частина виноситься в базові класи.

Моделі, навпаки, дуже товсті і містять велику частину коду, пов'язану з обробкою даних, тому що структура даних і бізнес-логіка, що міститься в них, зазвичай досить специфічна для конкретного додатка.

Для наглядності, на рисунку зображена діаграма класів.

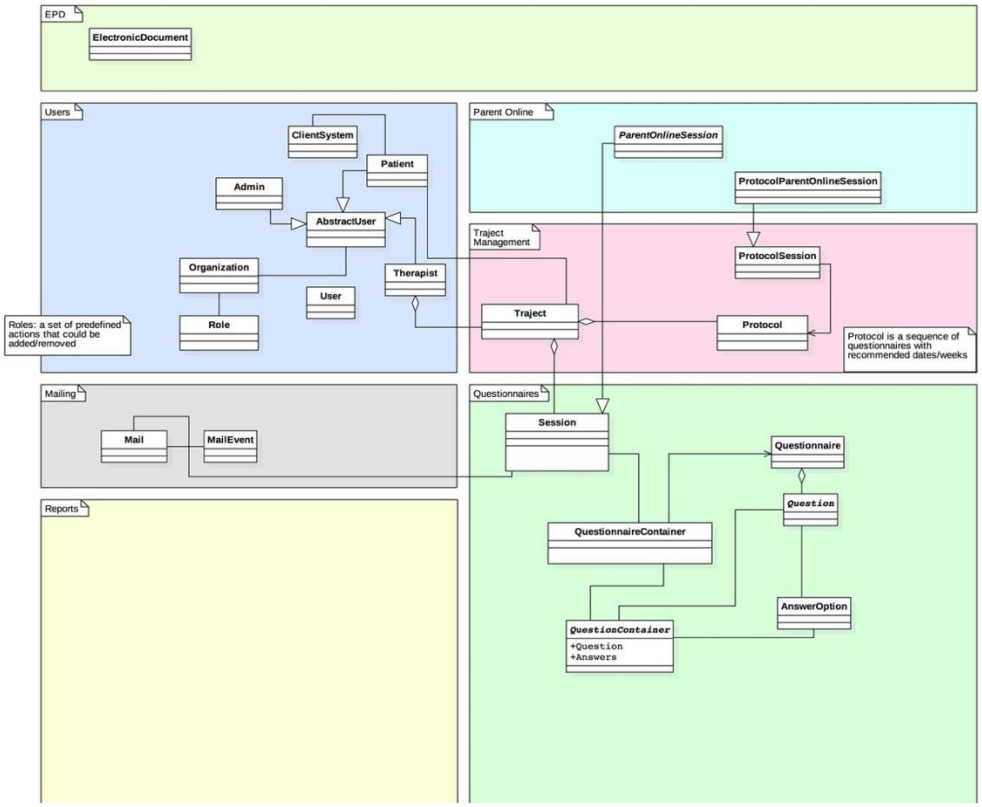


Рисунок 3.1 - Діаграма класів

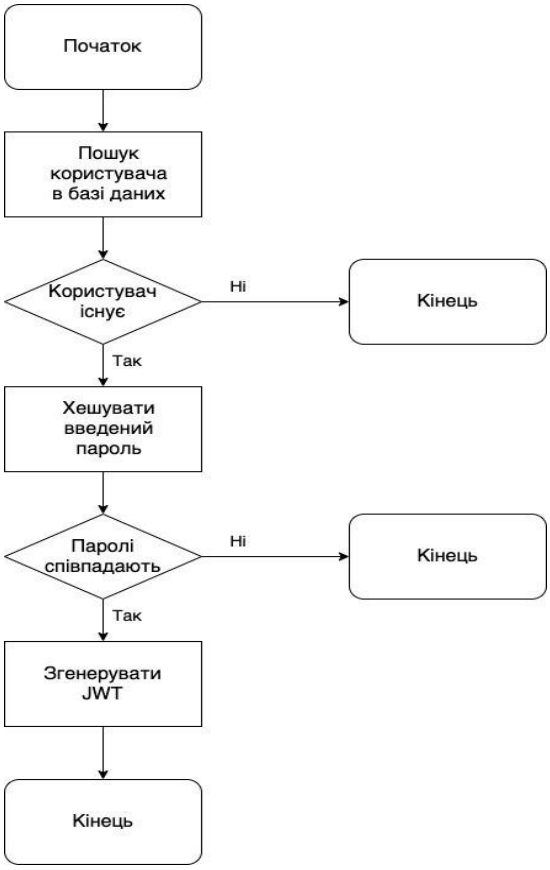


Рис. 3.3.1. Алгоритм аутентифікації користувача

ВИСНОВКИ ДО РОЗДІЛУ 3

Поєднання зазначених технологій для створення додатку вирішують проблеми кроссплатформенності, моніторингу здоров’я пацієнта та комунікацію з тепаревтом.

У третьому розділі були розглянуті технології, за допомогою яких можна створити веб-додаток. Всі варіанти було проаналізовано, були виявлені переваги та недоліки в кожному з них. Обґрунтований вибір мови програмування, платформи для додатку, враховуючи вимоги до функціоналу програмного продукту. Були розглянуті бібліотеки та фреймворки, які можуть допомогти в розробці веб-додатку. Серед них було обрано найбільш доцільний варіант – бібліотеку React.

Були описані основні рішення щодо реалізації проєкту з урахуванням використання обраної бібліотеки React. Спроєктовано систему зручного управління станами веб-додатку, яка досягається за допомогою бібліотеки Redux.

РОЗДІЛ 4
ОПИС ІНТЕРФЕЙСУ ДОДАТКА

4.1 Створення кроссплатформенного додатку

Перейдемо до безпосереднього візуально опису функціоналу інтерфейсу користувача (UI), і проведемо наочну демонстрацію основних елементів з їх описом.

Першой задачею було створення кроссплатформенного додатку, яким можна користуватися незалежно від платформи на якій він встановлений.

Для встановлення додатку потрібно перейти за посиланням в браузері та праворуч від пошуку натиснути кнопку ‘Install’.

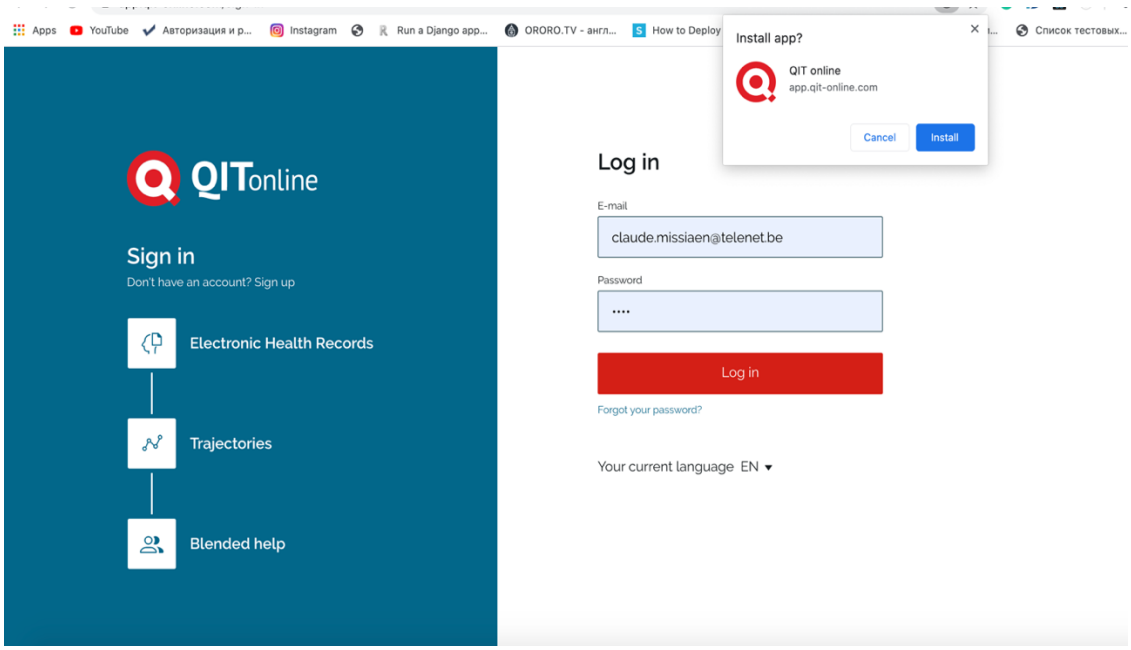


Рис. 4.1 Встановлення додатка на десктоп

Після цього можна користуватися додатком як десктопним.

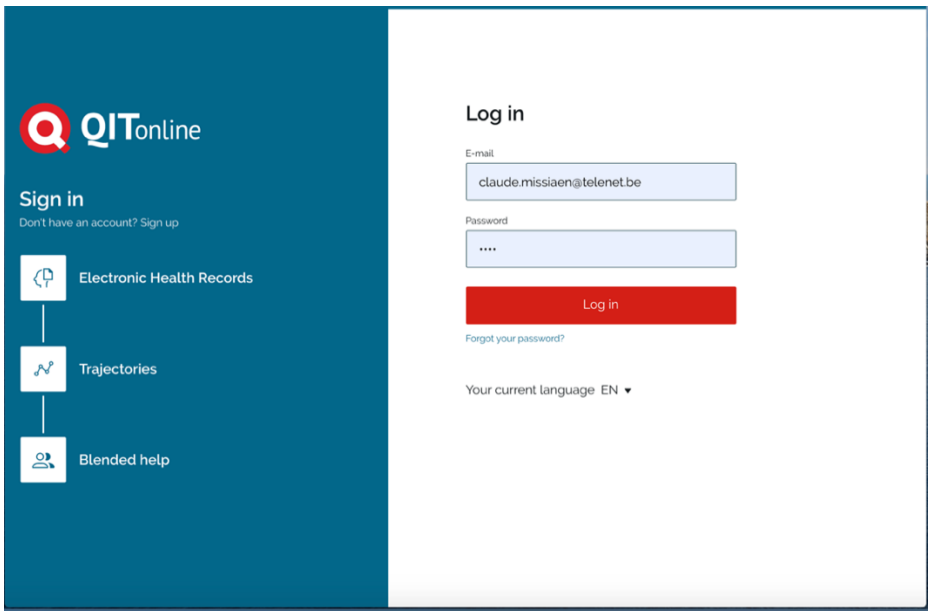


Рис. 4.2 Встановлений додаток

Для встановлення додатку на телефон з платформою Android потрібно в браузері перейти за адресою app.qit-online.com та натиснути кнопку “Добавить”.

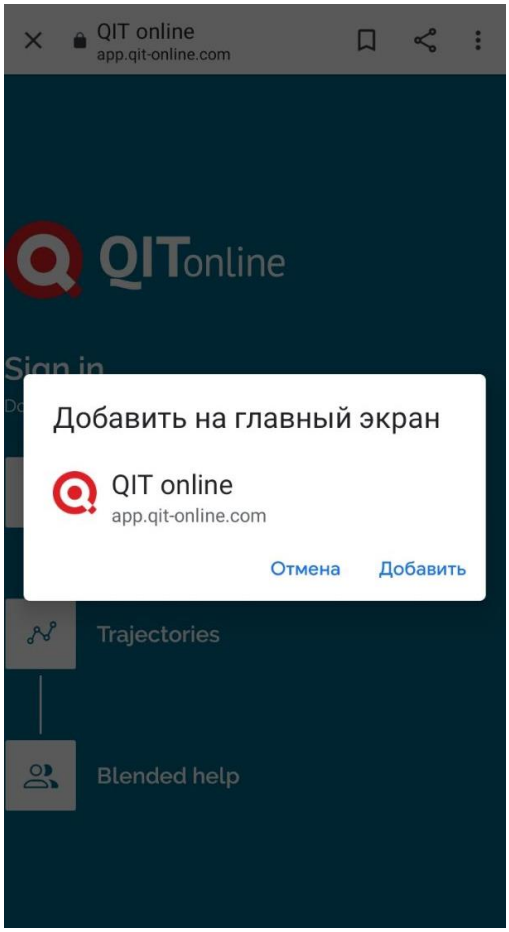


Рис. 4.3 Встановлення додатка на платформу Android

Для встановлення додатку на телефон з платформою iOS потрібно в браузері перейти за адресою app.qit-online.com, натиснути кнопку “Share”, а потім обрати “Add to Home Screen”

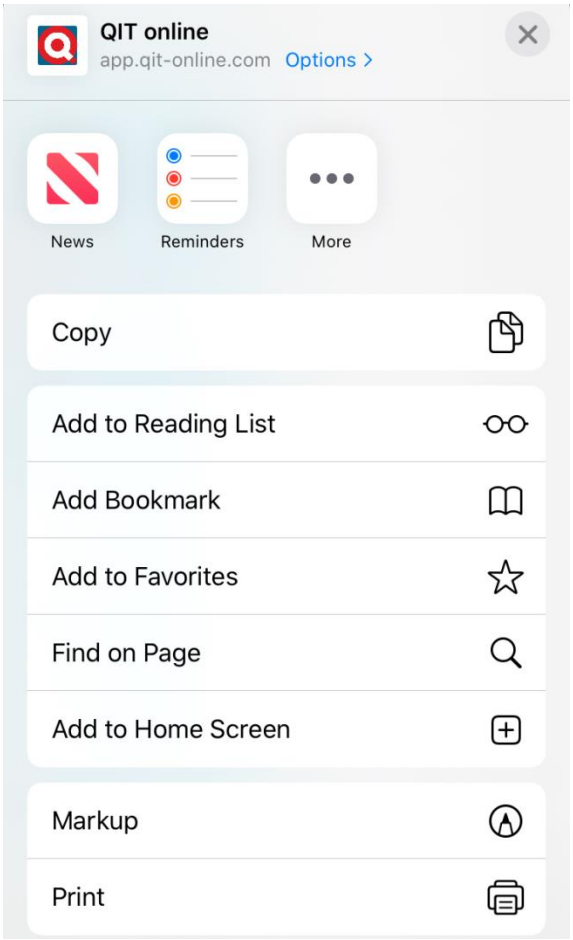


Рис. 4.4 Встановлення додатка на платформу iOS

Після цієї процедури на головний екран смартфону буде додана іконка з даним додатком, ним можна буде користуватися як нативним.

4.2 Процедура реєстрації користувача в додатку

Згідно із проєктованим дизайном було розроблено сторінки для процедури реєстрації.

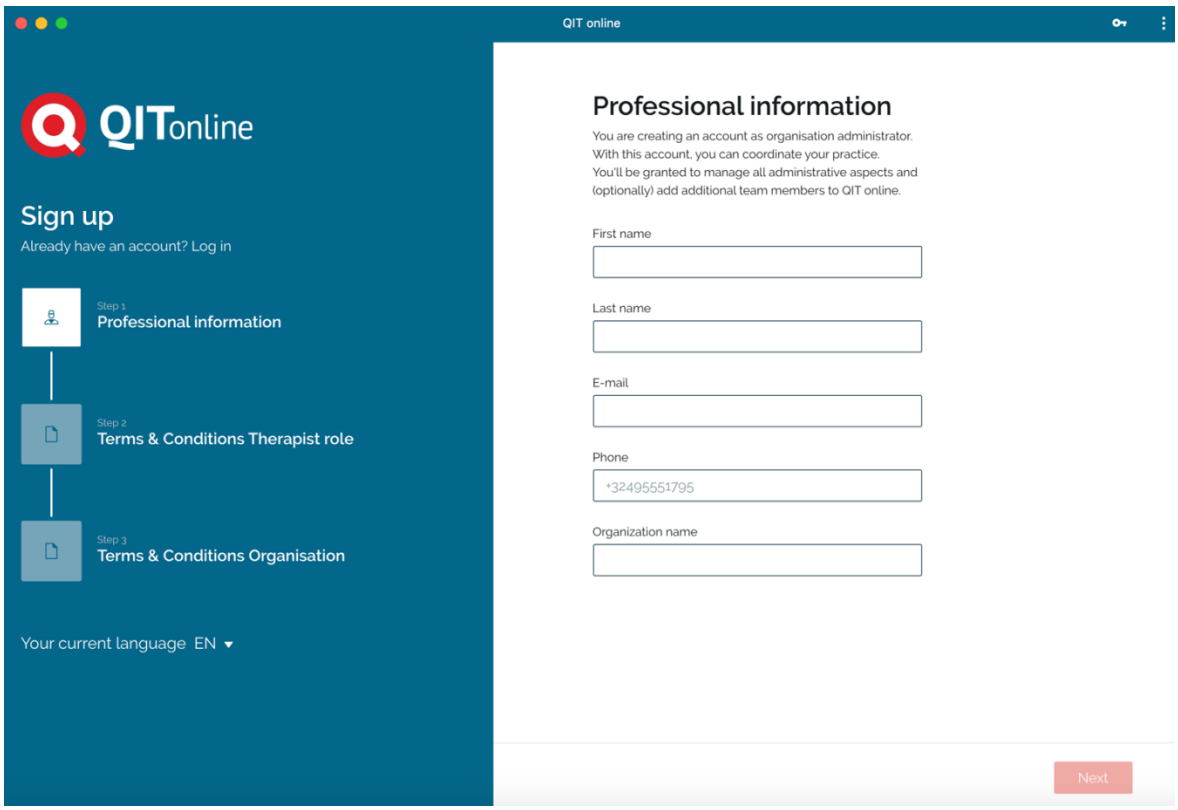


Рис. 4.5 Процес реєстрації

Після відправки форми, користувачу на пошту прийде посилання для створення паролю з тимчасовим токеном.

Після цього користувач може ввійти в систему ввівши логін та пароль.

Якщо пароль був втрачений чи забутий, його можна відновити ввівши пошту за допомогою якої користувач був зареєстрованим.

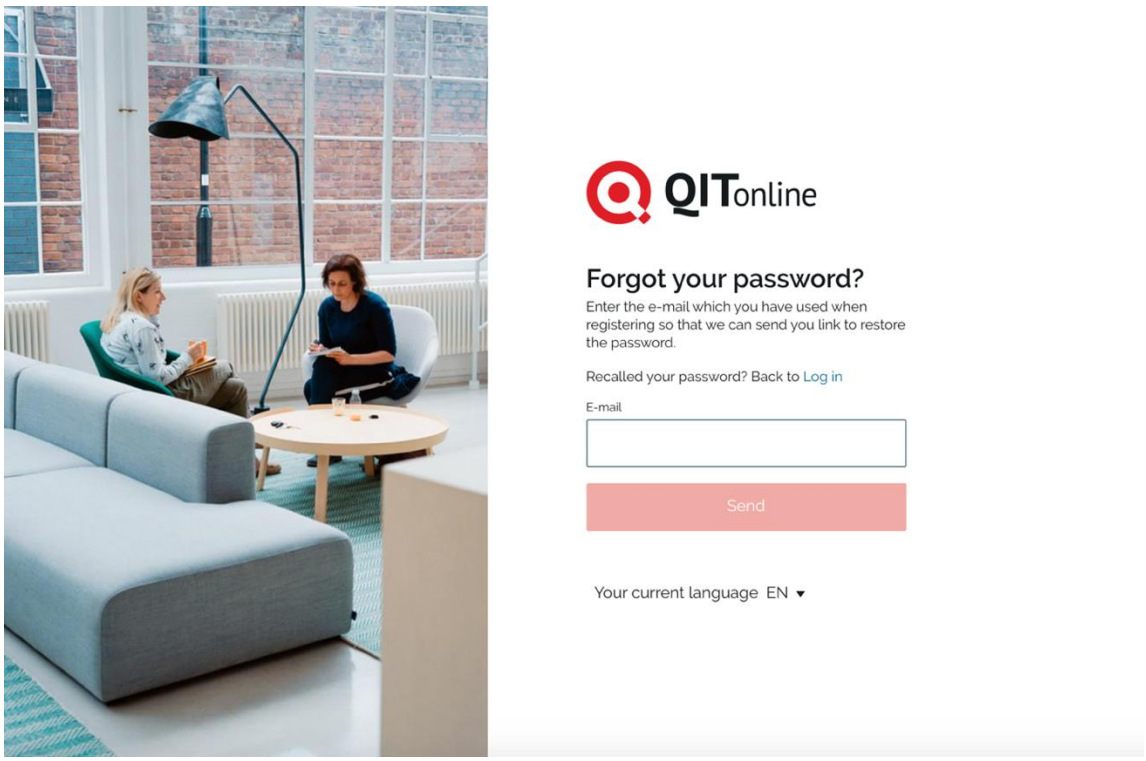
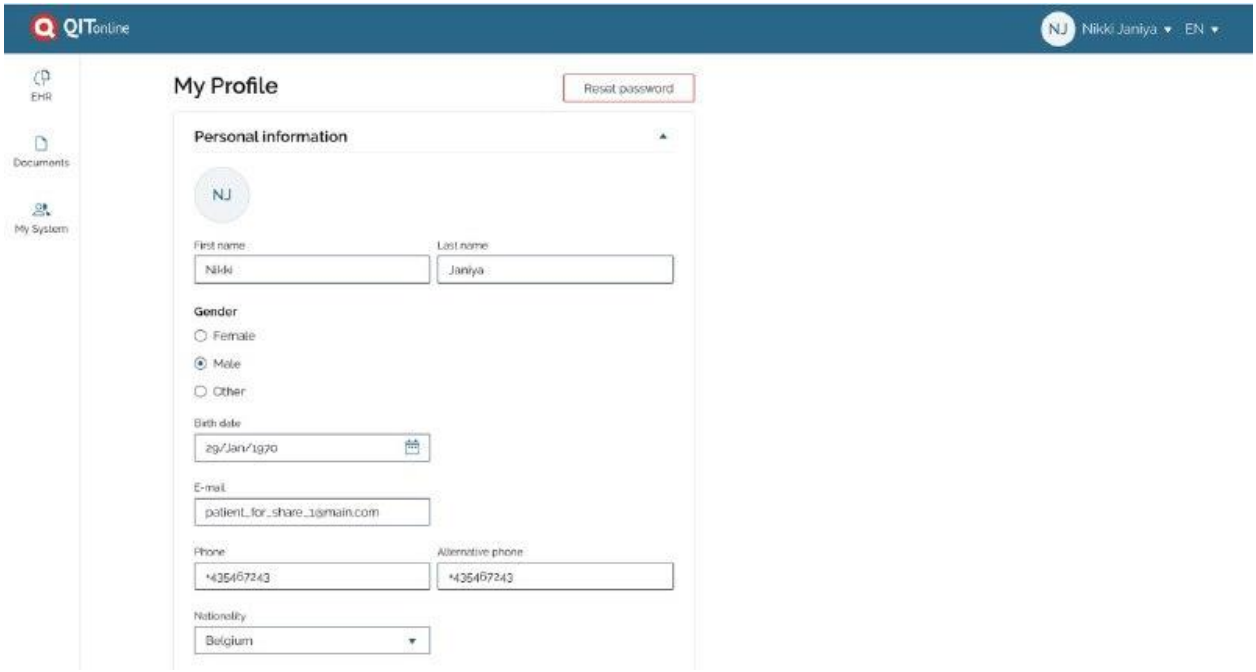


Рис. 4.6 Процес відновлення паролю

Бачимо, що на кожній сторінці реалізована зміна мови, що відповідає другій задачі “Інтернаціоналізація”.

Після введеня логіну та пароля, в залежності від ролі юзера, він потрапить в один із трьох інтерфейсів. Також кожен користувач може перейти на сторінку profile для редагування особистих даних: ім’я, прізвище, може завантажити фото, також змінити пароль натиснувиши кнопку “Reset password”



4.7 Інтерфейс особистого кабінету терапевта

Інтерфейс терапевта

Після входу в систему, терапевт потрапить на головну сторінку

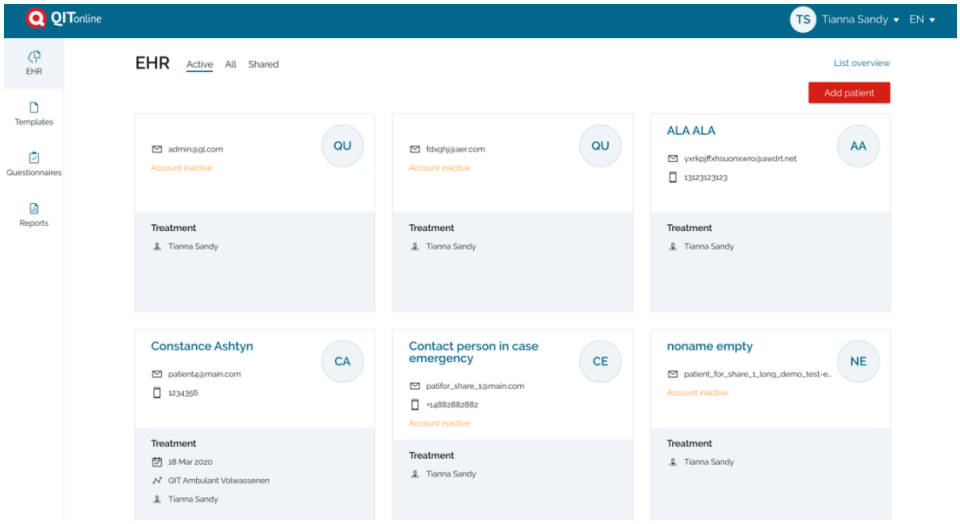


Рис. 4.8 Інтефейс терапевта

4.4 Процес створення облікового запису пацієнта

Терапевт має можливість створити нового пацієнта. Для цього потрібно

перейти на сторінку створення облікового запису пацієнта, заповнити поля форми та відправити. Після цього, на вказану пошту прийде посилання на сайт з унікальним токеном. Перейшовши за посиланням, та відправивши форму реєстраціє, пацієнт може ввійти в систему інтерфейсу пацієнта.

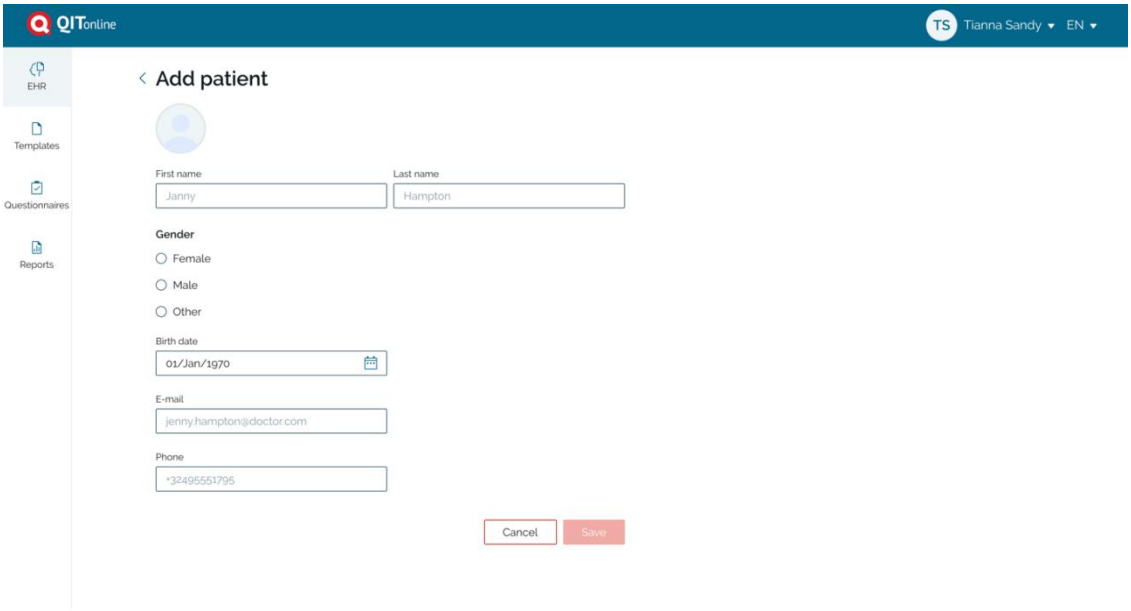


Рис. 4.9 Процес створення облікового запису пацієнта

Після процесу реєстрації нового користувача, терапевт може перейти на сторінку пацієнта, створити для нього нову траєкторію проходжування опитувальників, обмінятися документами.

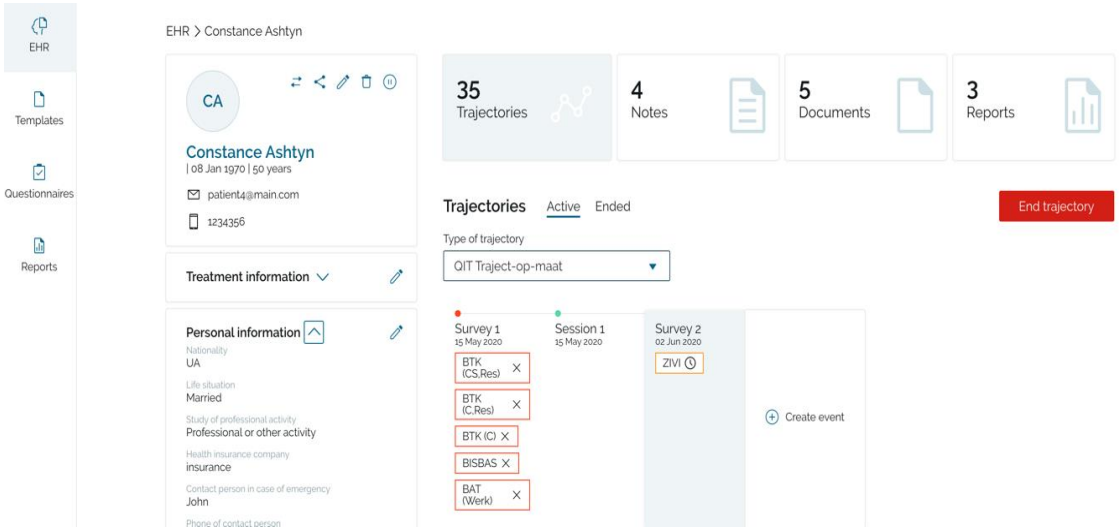


Рис. 4.10 Сторінка взаємодії з пацієнтом

Після проходженням пацієнтом опитувальника, на бекенді генерується репорт, який терапевт може переглянути або скачати в форматі pdf.

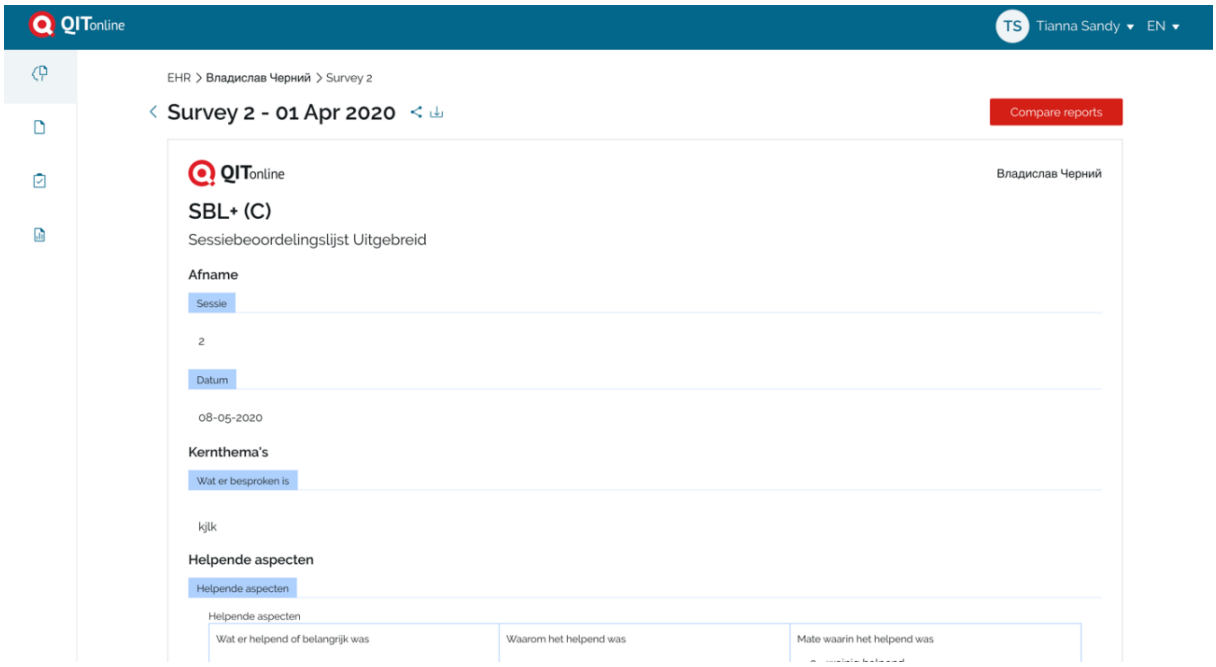


Рис. 4.11 Сторінка репорту

Після проходження процесу реєстрації, пацієнт має доступ до системи.

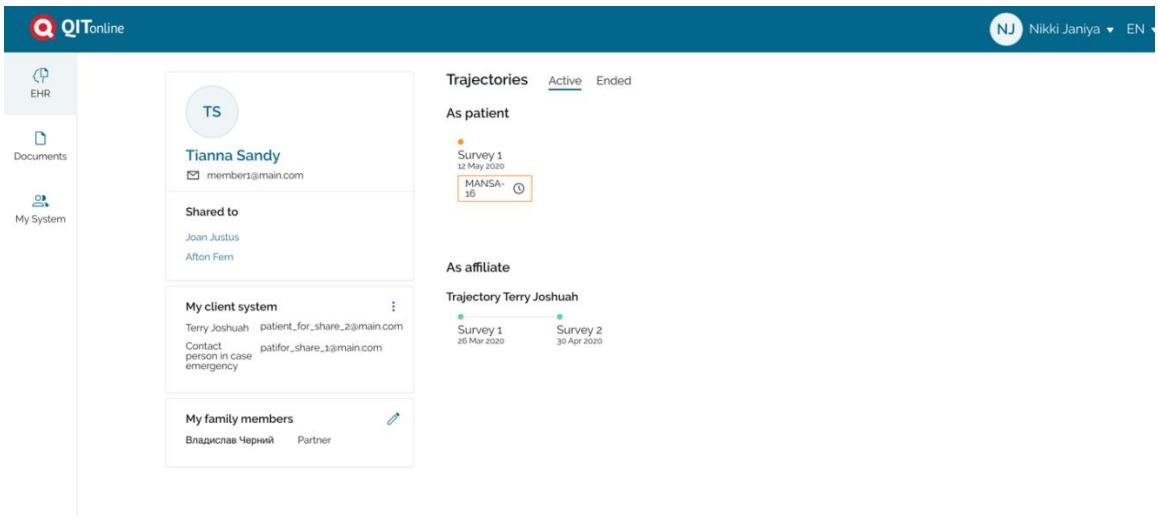


Рис. 4.12 Інтерфейс пацієнта

Пацієнт має змогу проходити опитувальники, які прив’язав до нього терепевт.

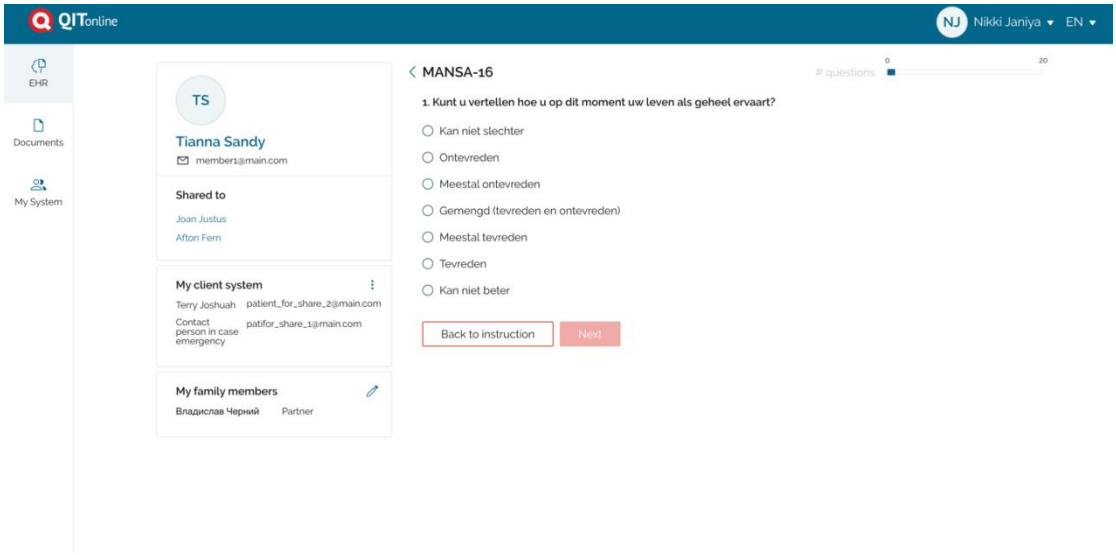


Рис. 4.13 Інтерфейс пацієнта

На вкладці “Документи” пацієнт може переглянути документи які завантажив терапевт, скачати файл, завантажити та поділитися файлом з терапевтом.

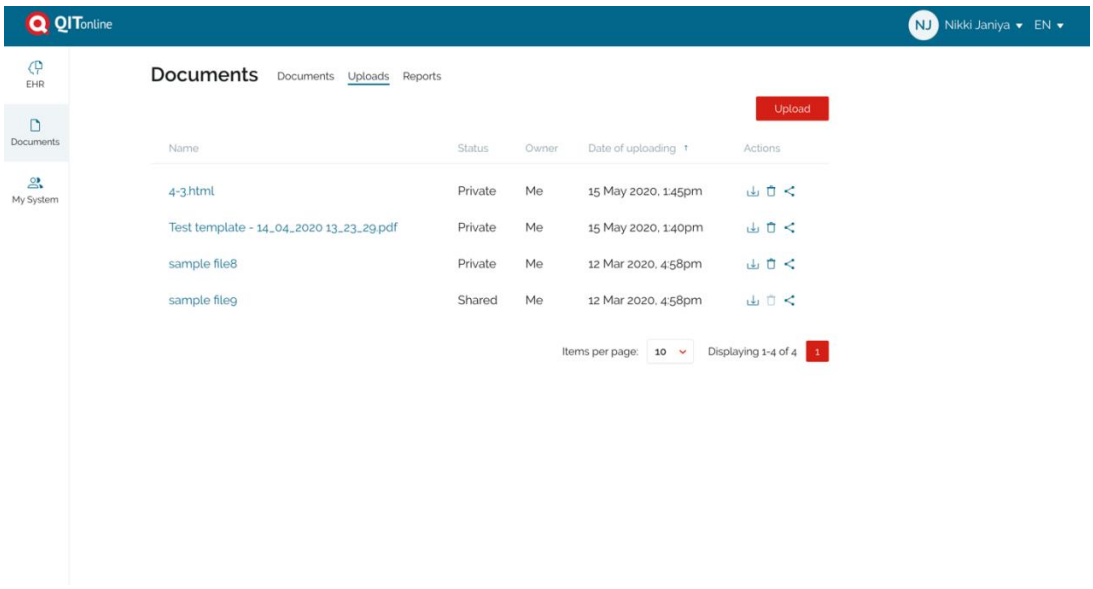


Рис. 4.14 Вкладка “Документи”

ВИСНОВКИ ДО РОЗДІЛУ 4

У даному розділі було описано результати експлутації розробленого програмного продукту. Детально було описано основні функції програми та їх призначення:

- Встановлення додатка на різні платформи.
- Реєстрація та вхід користувачів
- Створення пацієнтів.
- Механізми взаємодії з пацієнтом.
- Проходження опитувальників пацієнтом.
- Процес обміну документами

У процесі демонстрації було закріплено знімки екранів виконання програми, що наглядно демонструють функціональність додатку.

ВИСНОВКИ

Метою даного дипломного проєкту було створення додатку для моніторингу здоров’я пацієнта, який би був кроссплатформенним, спрощував документообіг лікарень та вирішував проблему комунікації пацієнта з лікуючим лікарем.

В ході виконання дипломної роботи була докладно вивчена теорія на тему створення веб-додатків, серверної і клієнтської його частин. Описано популярні фреймворки, які можуть використовуватися в створенні промислових додатків з високим ступенем завантаженості. Спираючись на вивчений матеріал, була розроблена серверна і клієнтська частина додатку, реалізований графічний інтерфейс.

Аналіз існуючих програмних рішень, попередньо виконаний в дипломному проєкті, показав доцільність створення даного проєкту. Для розроблення були використані відповідні технології, а саме: мова програмування PHP для реалізації серверної частини додатку, мова програмування JavaScript, бібліотека React, фреймворк Redux для клієнтської частини та JavaScript бібліотека Workbox для реалізації PWA (progressive web app). Зазначені засоби реалізації дозволили забезпечити стабільну роботу додатка.

Розроблений додаток:

1. Працює на будь якій платформі;
2. Підтримує інтернаціоналізацію;
3. Забезпечує моніторинг здоров’я пацієнта;
4. Спрощує документообмін між пацієнтом та терапевтом.

Отже, розроблення виконано у повному обсязі, відповідає всім вимогам, описаним в документі, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1. API [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/web-api/;

2. Протокол HTTP [Электронный ресурс]. – https://developer.mozilla.org/ru/docs/Web/HTTP;

3. Дэвид Флэнаган, «JavaScript. подробное руководство, 6-е издание», 2012 г.;

4. Стоян Стефанов «React.js Быстрый старт», 2017г.;

5. А. Бенкс, Е. Порселло, «React и Redux. Функциональная веб-разработка», 2018г.;

6. Руководство по React [Электронный ресурс]. – Режим доступа:

7. https://metanit.com/web/react/;

8. Руководство по React Router [Электронный ресурс]. – Режим доступа: https://re.acttraining.com/react-router/core/guides/quick-start;

9. Джон Дакет, «HTML и CSS. Разработка и дизайн веб-сайтов»;

10. Гаевский, А.Ю. 100% самоучитель. Создание Web-страниц и Web-сайтов. HTML и JavaScript / А.Ю. Гаевский, В.А. Романовский. - М.: Наука, 2015. - 464 с.

11. Дронов, В. JavaScript в Web-дизайне / В. Дронов. - М.: СПб: БХВ, 2014. - 880 с.

12. Кингсли-Хью, К.Э. JavaScript 1.5: учебный курс / К.Э. Кингсли-Хью. - М.: СПб: Питер, 2013. - 272 с.

13. Негрино JavaScript для начинающих / Негрино, Том. - М.: Огни, 2013. - 544 с.

14. Федоров, А.Г. JavaScript для всех / А.Г. Федоров. - М.: Машиностроение, 2012. - 384 с.

15. Билл Кеннеди, Чак Муссиано - "HTML и XHTML. Подробное руководство (HTML & HXTML. The Definitive Guide)"

16. Эрик Мейер - "CSS-каскадные таблицы стилей. Подробное руководство (Cascading Style Sheets: The Definitive Guide)"

17. К. Шмитт - "CSS. Рецепты программирования (CSS: Cookbook)"

18. Б. Хеник - "HTML и CSS. Путь к совершенству (HTML и CSS: The Good Parts)"

19. Блинов, И.Н. Java. Промышленное программирование : практ. пособие

20. / И.Н. Блинов, В.С. Романчик. –Минск : УниверсалПресс, 2007. – 704 с.

21. Офіційний сайт Icarus Verilog – Режим доступа :

22. http://iverilog.icarus.com/ – Дата доступу: 07.05.2020.

23. Анопрієнко О. Я. Методика проектування Web-додатків з використанням платформи Java EE / Анопрієнко О. Я. // Інформатика і компютерні технології – 2010 – № VI – С. 160 - 165 .

24. Хорстманн, Кей С., Корнелл Гари . Java2. Основы, 7-е изд./

25. Хорстманн, Кей С., Корнелл Гари: Пер. с англ. – М.: И.Д. «Вильямс», 2006. – 896 с.

26. Budi Kurniawan . Java for the Web with Servlets, JSP, and EJB: A Developer's Guide to J2EE Solutions / Budi Kurniawan : New Riders Publishing , April 12 2002 – 976 p.

27. Э. Фримен, Э. Фримен. Изучаем HTML, XHTML и CSS = Head First HTML with CSS & XHTML. — П.: «Питер», 2010. — 656 с.

28. Стивен Шафер. HTML, XHTML и CSS. Библия пользователя, 5-е издание = HTML, XHTML, and CSS Bible, 5th Edition. — с.56.

29. Дэвид Макфарланд. Большая книга CSS3 – CSS3. СПБ.: Питер,2016 – 251 с.

30. Бер Бибо, Иегуда Кац. jQuery. Подробное руководство по продвинутому JavaScript – СПБ:Питер.2011. – 258 с.

31. Бретт Маклафлин. Объектно – ориентированный анализ и проектирование – С.: O’Reilly.258 с.

32. Лиза Гарднер, Джейсон Григсби. Разработка веб-сайтов для мобильных устройств – М.:Вильямс,2014. – 268 с.

33. Дэвид Сойер Макфарланд.Разработка сайтов на JavaScript и jQuery. Исчерпывающее руководство – С.: O'Reilly,2012. – 587 с.

34. Эрик Фримен, Элизабет Робсон. Head First JavaScript Programming – С.: O'Reilly,2010. – 267 с.

35. Кэти Сиерра, Берт Бейтс.Head First Java – O'Reilly.2014. –155 с.

36. Роберт Лафоре.Структуры данных и алгоритмы в Java – С.: O'Reilly,2010. – 55 с.

37. А.Аллан.Программирование для мобильных устройств – СПБ.: O'Reilly.2012. – 188с.

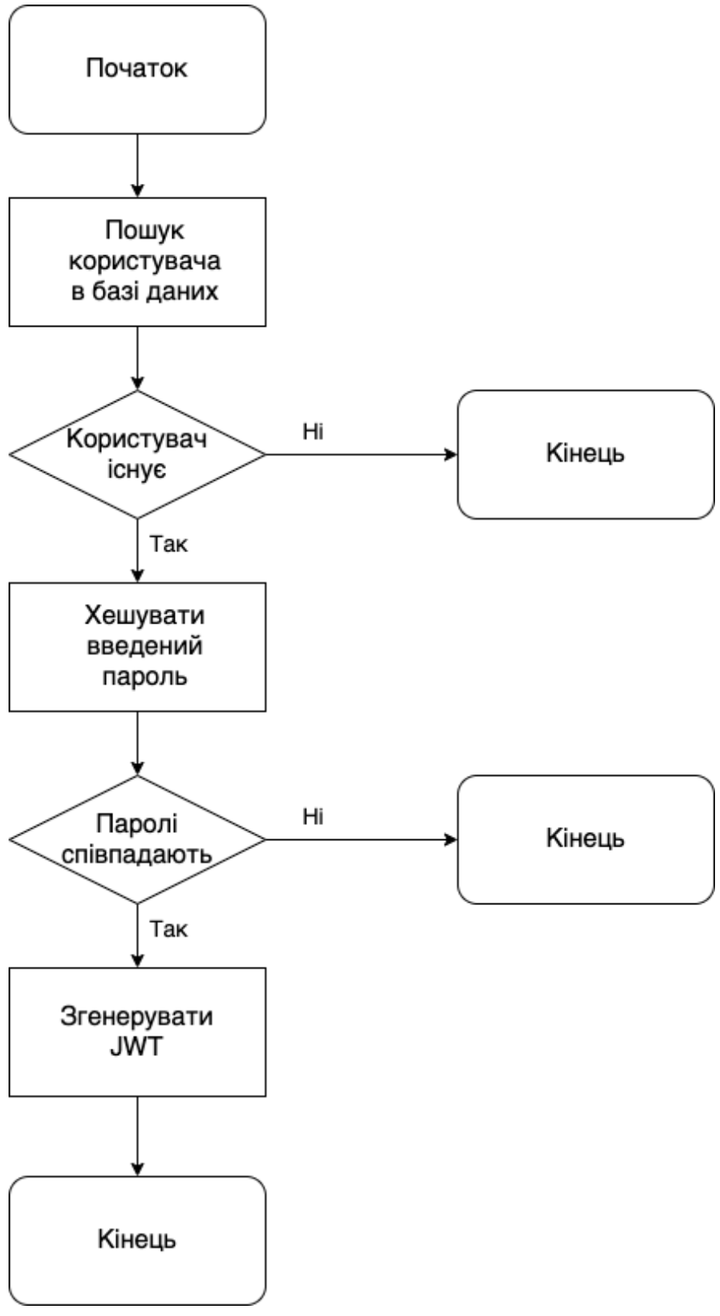
38. Скляр Д., Трахтенберг А..PHP. Рецепты программирования. 3-изд. – СПБ.:Питер.2012. – 335с.

39. А.Аллан. Клиентская разработка для профессионалов. Node.js – СПБ.: Питер.2017. – 220с

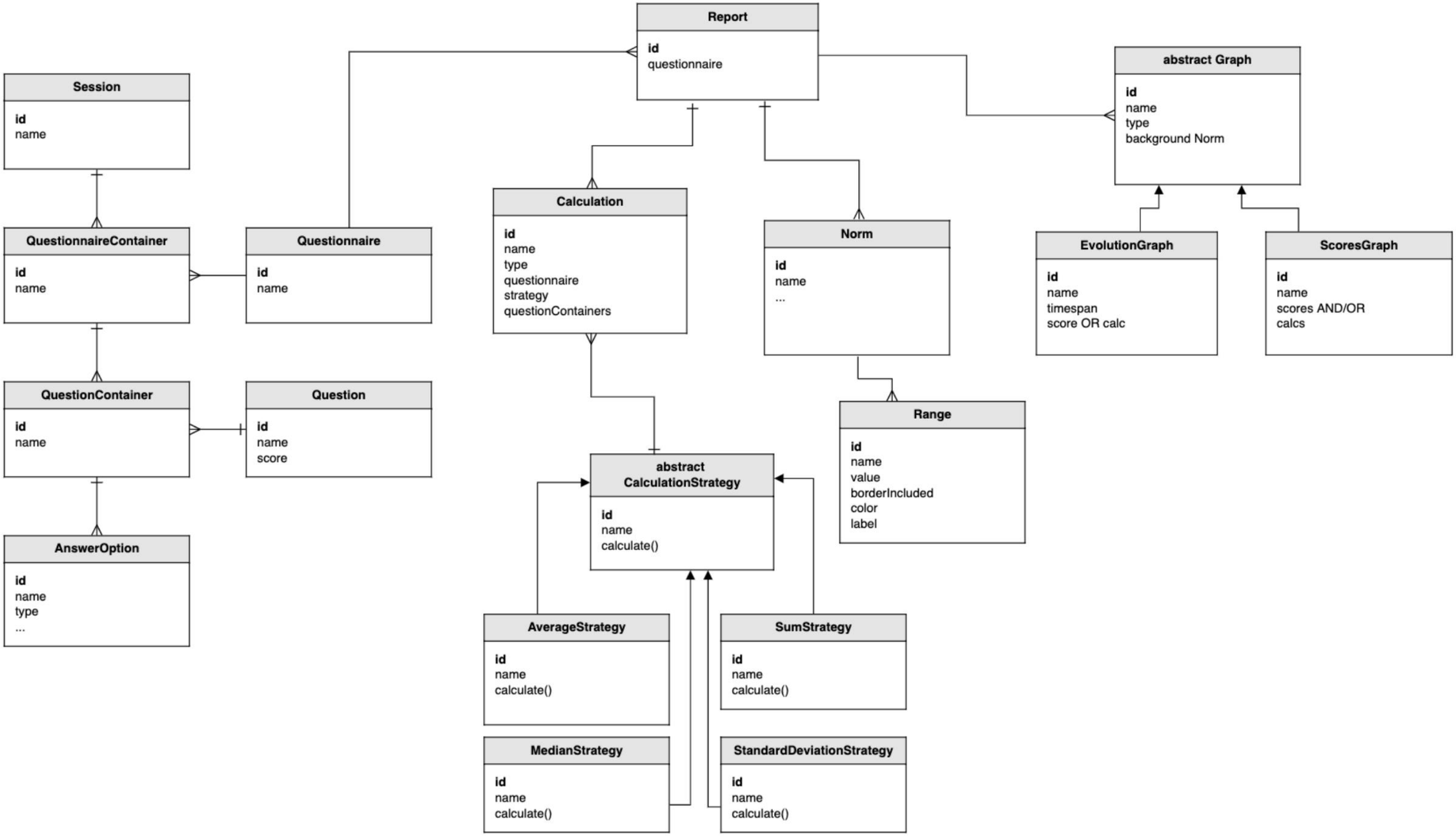
40. Автоматизація [Електронний ресурс]. – Точка доступу: URL: http://uk.wikipedia.org/wiki/Автоматизація – Автоматизація.

41. Веб застосунок [Електронний ресурс]. – Точка доступу: URL: http://uk.wikipedia.org/wiki/Веб-застосунок– Веб застосунок.

ДОДАТОК А



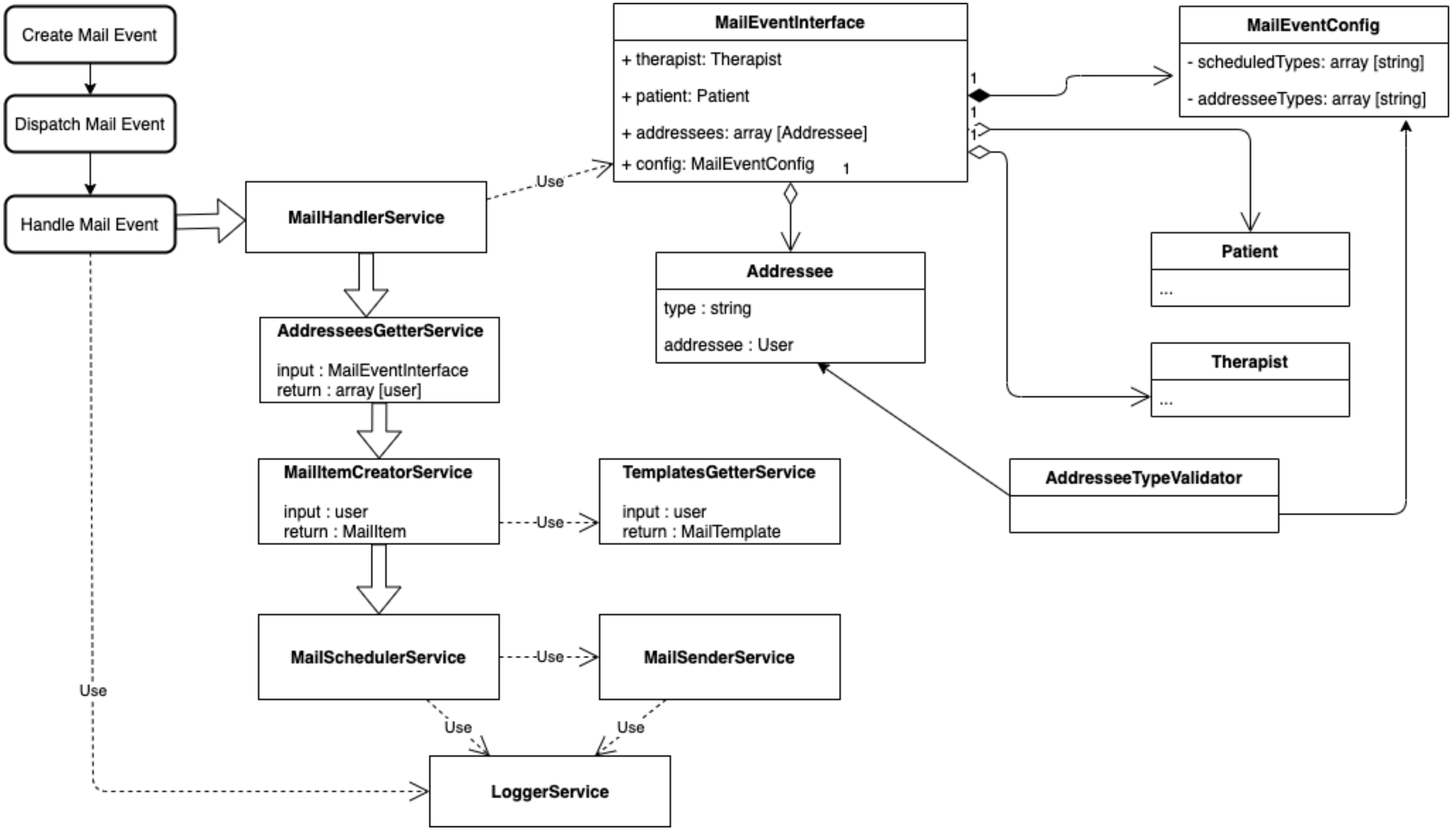
					ДП 6309. 03.000 Д1	
Зм	Ар	№	Підпи	Дата	Блок схема алгоритму аутентифікації користувача	Літера / Маса / Масшта
Розроб.		Голота М. М.				
Перевір		Антонюк А.І.				Аркуш / Аркушів 1
Т. контр.						НТУУ "КПІ" ФІОТ
Н. контр.		Сімоменко В.П.				Група ІО-63
Затв.						



					ДП 6309. 04.000 Д2
					Структура сховища даних
					Схема функціональна
Зм.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Голота М.М			
Перевір.		Антонюк А.І.			
Т. контр.					
Н. контр.		Сімоменко В.П.			
Затв.					

Літера	Маса	Масштаб
Аркуш 1	Аркушів 1	

НТУУ "КПІ" ФІОТ
Група ІО-63



					ДП 6309. 05.000 ДЗ
Зм.	Арк.	№ докум.	Підпис	Дата	Взаємодія класів в системі Діаграма класів
Розроб.		Голота М.М			Літера / Маса / Масштаб
Перевір.		Антонюк А.І.			Аркуш 1 / Аркушів 1
Т. контр.					НТУУ "КПІ" ФІОТ Група ІО-63
Н. контр.		Сімоменко В. І.			
Затв.					

ДОДАТОК Б

```
{
"name": "qit",
"version": "2.0.0",
"description": "Qit frontend application",
"repository": "git@github.com:GolotaMishka/qit-app.git",
"author": "Panenco <info@panenco.com>",
"license": "UNLICENSED",
"private": true,
"scripts": {
"start": "npm run server start",
"lerna": "lerna",
"build": "yarn lerna run build && yarn ui build:storybook",
"bootstrap": "lerna bootstrap",
"ui": "cd packages/ui && yarn",
"app": "cd packages/app && yarn",
"data": "cd packages/data && yarn",
"server": "cd packages/server && npm run",
"cypress": "cd packages/app && cypress open",
"cypress-run": "cd packages/app && cypress run",
"heroku-prebuild": "export NPM_CONFIG_PRODUCTION=false; export
YARN_PRODUCTION=false; yarn",
"heroku-postbuild": "npm run build && export
NPM_CONFIG_PRODUCTION=true; export YARN_PRODUCTION=true;",
"upload:ui:staging": "npm run ui build:storybook && scp -r ./packages/ui/static-
story/* root@104.248.93.6:/var/www/html/ui",
"upload:app:staging": "lerna run clean && cross-env PROJECT_ENV=staging
NODE_ENV=production npm run build && scp -r ./packages/app/dist/*
root@104.248.93.6:/var/www/html",
"upload:app:production": "lerna run clean && cross-env PROJECT_ENV=production
NODE_ENV=production npm run build && scp -r ./packages/app/dist/*
root@134.209.196.208:/var/www/html"
},
"husky": {
"hooks": {
"commit-msg": "commitlint -E HUSKY_GIT_PARAMS"
}
},
"
"packages/**"
]
}
}
```

```
const App = () => {
return (
<Provider store={store}>
<PersistGate loading={<Loader />} persistor={persistor}>
<Router history={history}>
<AuthProvider>
<Qit />
</AuthProvider>
</Router>
</PersistGate>
</Provider>
);
};

fetch(svgSprite)
.then(response => response.text())
.then(svg => {
const dummyElement = document.createElement('div');
dummyElement.innerHTML = svg;
const svgElement = dummyElement.firstElementChild;
svgElement.style.display = 'none';
document.body.prepend(svgElement);
})
.catch(error => {
console.error(error);
});

if (process.env.NODE_ENV === 'production' && 'serviceWorker' in navigator) {
window.addEventListener('load', () => {
const wb = new Workbox('/service-worker.js', { scope: '/' });
wb.register();
});
}

render(<App />, rootElement);
```

```
/* global self */

import { precacheAndRoute, createHandlerBoundToURL } from 'workbox-precaching';
import { registerRoute, NavigationRoute, setCatchHandler } from 'workbox-routing';
import { NetworkOnly, StaleWhileRevalidate, CacheFirst, NetworkFirst } from 'workbox-
strategies';
import { ExpirationPlugin } from 'workbox-expiration';
import { BackgroundSyncPlugin } from 'workbox-background-sync';
import { CacheableResponsePlugin } from 'workbox-cacheable-response';

/* eslint-disable-next-line */
precacheAndRoute(self.__WB_MANIFEST);

const handler = createHandlerBoundToURL('/index.html');
const navigationRoute = new NavigationRoute(handler);
registerRoute(navigationRoute);

const apiRegExp = new RegExp(`^https://api.qit-online.com/api/v1/`);

registerRoute(
apiRegExp,
new NetworkFirst({
cacheName: 'qit-api-calls',
plugins: [
new ExpirationPlugin({
maxAgeSeconds: 30 * 60, // 30 minutes
}),
],
}),
);

setCatchHandler(new NetworkOnly());

const bgSyncPlugin = new BackgroundSyncPlugin('my-qit-interaction', {
maxRetentionTime: 24 * 60, // Retry for max of 24 Hours (specified in minutes)
});

registerRoute(
apiRegExp,
new NetworkOnly({
plugins: [bgSyncPlugin],
}),
'POST',
);
```

```
registerRoute(
apiRegExp,
new NetworkOnly({
plugins: [bgSyncPlugin],
}),
'PUT',
);

registerRoute(
apiRegExp,
new NetworkOnly({
plugins: [bgSyncPlugin],
}),
'PATCH',
);

// Cache the Google Fonts stylesheets with a stale-while-revalidate strategy.
registerRoute(
/^https:\/\/fonts\.googleapis\.com/,
new StaleWhileRevalidate({
cacheName: 'google-fonts-stylesheets',
}),
);

// Cache the underlying font files with a cache-first strategy for 1 year.
registerRoute(
/^https:\/\/fonts\.gstatic\.com/,
new CacheFirst({
cacheName: 'google-fonts-webfonts',
plugins: [
new CacheableResponsePlugin({
statuses: [0, 200],
}),
new ExpirationPlugin({
maxAgeSeconds: 60 * 60 * 24 * 365,
maxEntries: 30,
}),
],
}),
);
```

```
const AuthorizedAdminRoute = withAuth({
redirectTo: signInPath,
role: Authorized.Roles.ROLE_SUPER_ADMIN,
preloader: <Loader />,
})(Admin);

const AuthorizedTherapistRoute = withAuth({
redirectTo: signInPath,
role: Authorized.Roles.ROLE_THERAPIST,
preloader: <Loader />,
})(Therapist);

const AuthorizedPatientRoute = withAuth({
redirectTo: signInPath,
role: Authorized.Roles.ROLE_PATIENT,
preloader: <Loader />,
})(Patient);

class Qit extends React.Component {
componentDidMount() {}

componentDidUpdate() {}

render() {
if (process.env.PROJECT_ENV === 'production') {
ReactGA.initialize('UA-89464105-12');
TagManager.initialize({ gtmId: 'GTM-P6KH8BH' });

ReactGA.pageview(window.location.pathname + window.location.search);
}

return (
<Switch>
<Route path={baseAdminPath} component={AuthorizedAdminRoute} />
<Route path={baseTherapistPath} component={AuthorizedTherapistRoute} />
<Route path={basePatientPath} component={AuthorizedPatientRoute} />
<Route path={basePath} component={Shared} />
</Switch>
);
}
}
export default hot(module)(Qit);
```

```
import AddPatientPage from 'components/app/therapist/epd/add-patient';
import { format } from 'date-fns';

const mapStateToProps = state => ({
currentUserId: selectors.auth.getSessionUserId(state),
});

const mapDispatchToProps = {
add: actions.therapists.createPatient,
};

class AddPatient extends React.Component {
static propTypes = {
history: RouterPropTypes.history.isRequired,
add: PropTypes.func.isRequired,
};

static defaultProps = {};

onSubmit = async (values, formikActions) => {
const newValues = {
...values,
birthday: format(values.birthday, 'DD/MM/YYYY'),
};
const userId = await this.props.add(newValues, formikActions);
if (userId) {
this.props.history.push(epdPatientPath(userId));
}
};

render() {
return (
<Formik
enableReinitialize
validationSchema={validation.PatientSchema}
onSubmit={this.onSubmit}
component={AddPatientPage}
/>
);
}
}
```

```
import React from 'react';
import PropTypes from 'prop-types';
import { RouterPropTypes } from 'utils/router-proptypes';
import { Formik } from 'formik';
import { Breadcrumbs } from '@qit/ui';
import { Trans } from 'react-i18next';
import withPaginatedSelect from 'utils/with-paginated-select';

import { epdPatientDocumentsPath, epdPatientDocumentsTemplatesAdd } from
'constants/url/therapist';
import AddTemplateComponent from
'components/app/therapist/epd/page/files/documents/templates/add-template';
import { connect, actions, selectors } from '@qit/data';

const mapStateToProps = state => ({
additionalParams: {
therapist: selectors.auth.getSessionUserId(state),
},
list: selectors.templates.getItemsList(state),
listLoaded: selectors.templates.getItemsListLoaded(state),
listTotal: selectors.templates.getItemsTotal(state),
});

const mapDispatchToProps = {
fetchList: actions.templates.fetchList,
add: actions.therapists.addFromTemplates,
};

class AddDocumentsRoute extends React.Component {
static propTypes = {
match: RouterPropTypes.match.isRequired,
history: RouterPropTypes.history.isRequired,
add: PropTypes.func.isRequired,
};

constructor(props) {

super(props);
this.state = {
template: null,
};
}

setTemplate = template => {
this.setState({
template,
});
};
```

```
onSubmit = async (values, formikActions) => {
if (!this.state.template) {
formikActions.setSubmitting(false);
formikActions.setErrors({ title: 'Required' });
} else {
const newValues = {
...values,
title: this.state.template.title,
};

if (await this.props.add(this.props.match.params.patientId, this.state.template.id, newValues,
formikActions)) {
await this.props.history.push(epdPatientDocumentsPath(this.props.match.params.patientId));
}
}
};

export const getInitialCredentials = () => (dispatch, getState, { api }) => {
const cookies = Cookies.getJSON('auth');
const cookiesCookie = Cookies.getJSON('cookies');
if (cookies) {
api.setAuthorization(cookies.auth);
}

if (cookies && cookiesCookie) {
dispatch({
type: constants.LOGIN_SUCCESS,
payload: {
didAcceptCookies: cookiesCookie.didAcceptCookies,
access: cookies.auth,
},
});
} else if (cookiesCookie && cookiesCookie.didAcceptCookies) {
return dispatch({ type: constants.COOKIES });
}
};
export const acceptCookies = () => dispatch => {
Cookies.set('cookies', {
didAcceptCookies: true,
});
return dispatch({ type: constants.COOKIES });
};
```

```
export const fetchSession = (credentials, formik) => async (dispatch, getState, { api }) => {
dispatch({ type: constants.LOGIN });
// formik.setSubmitting(true);
const cookiesCookie = Cookies.getJSON('cookies');

try {
const res = await api.post('/sign-in?version=v1', credentials);

api.setAuthorization(res.data.token);
Cookies.set('auth', {
auth: res.data.token,
});

// formik.setSubmitting(false);
dispatch({
type: constants.LOGIN_SUCCESS,
payload: {
didAcceptCookies: cookiesCookie ? cookiesCookie.didAcceptCookies : false,
access: res.data.token,
},
});
return true;
} catch (e) {
dispatch({ type: constants.LOGIN_FAILED });
// formik.setSubmitting(false);
formik.resetForm();
if (e.response && e.response.data) {
formik.setErrors(e.response.data.message);
}
return false;
}
};
export const logout = () => (dispatch, _, { api }) => {
api.removeAuthorization();
Cookies.remove('auth', { path: '/' });
return dispatch({ type: constants.LOGOUT });
};
```

```
import axios from 'axios';
import { HTTPError /* , ConnectionError */ } from 'utils/errors';
import * as apiConstants from 'constants/api';
import * as authConstants from 'constants/auth';
import * as authSelectors from 'selectors/auth';

const wait = time => new Promise(res => setTimeout(res, time));

class API {
constructor(store, ...axiosArgs) {
this.refreshInApi = this.refreshInApi.bind(this);
this.errorInterceptor = this.errorInterceptor.bind(this);
this.successInterceptor = this.successInterceptor.bind(this);
this.setAuthorization = this.setAuthorization.bind(this);
this.removeAuthorization = this.removeAuthorization.bind(this);

this.axios = axios.create(...axiosArgs);
this.store = store;
this.axios.interceptors.response.use(this.successInterceptor, this.errorInterceptor);

if (window) window.api = this;
}

static noRefresh = 'No token';

async refreshInApi() {
const refresh = authSelectors.getRefresh(this.store.getState());
if (!refresh) throw new Error(API.noRefresh);
const res = await this.axios.post('/api/token/refresh/', { refresh });
this.setAuthorization(res.data.access);
this.store.dispatch({
type: authConstants.REFRESH_SUCCESS,
payload: {
...res.data,
},
});

return res.data;
}

successInterceptor(response) {
this.store.dispatch({ type: apiConstants.SUCCESS });
return response;
}
```

```
errorInterceptor(error) {
if (error.response) {
switch (error.response.status) {
case 400:
if (error.config) {
const reqConfig = error.config;
if (reqConfig.url === `${reqConfig.baseURL}/api/token/refresh/`) {
this.store.dispatch({
type: authConstants.LOGOUT,
});
}
}
break;
case 401:
this.store.dispatch({
type: authConstants.LOGOUT,
});
break;
case 500:
this.store.dispatch({ type: apiConstants.SERVER_ERROR });
throw new HTTPError('Server Error', error.response.status);
case 404:
this.store.dispatch({ type: apiConstants.NOT_FOUND });
throw new HTTPError('Not Found', error.response.status);
default:
break;
}
} else {
this.store.dispatch({ type: apiConstants.CONNECTION_ERROR });
// throw new ConnectionError('No connection');
if (error.config) {
// const reqConfig = error.config;

console.log('Connection error, will retry in 3 sec');

return wait(3000).then(() => Promise.reject(error) /* this.axios.request(reqConfig) */);
}
}
return Promise.reject(error);
}
```

```
setAuthorization(access) {
this.axios.defaults.headers.common.Authorization = `Bearer ${access}`;
}

setRefresh(refresh) {
this.axios.defaults.headers.common['x-refresh-token'] = refresh;
}

removeAuthorization() {
delete this.axios.defaults.headers.common['x-auth'];
delete this.axios.defaults.headers.common['x-refresh-token'];
}

get = (...args) => this.axios.get.apply(this, args);

post = (...args) => this.axios.post.apply(this, args);

patch = (...args) => this.axios.patch.apply(this, args);

put = (...args) => this.axios.put.apply(this, args);

delete = (...args) => this.axios.delete.apply(this, args);
}

export default API;
import { createStore as reduxCreateStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';

import { composeWithDevTools } from 'redux-devtools-extension/developmentOnly';
import Immutable, { isImmutable } from 'immutable';
import { persistStore, createTransform, persistReducer } from 'redux-persist';
import createFilter from 'redux-persist-transform-filter-immutable';
import Serialize from 'remotedev-serialize';
import storage from 'redux-persist/lib/storage';

import { baseURL } from 'config';
import API from 'api';
import * as schema from 'utils/schemas';
import { normalize } from 'normalizr';
import reducers, { entities } from 'reducers';

/* eslint-disable */
export let api;
/* eslint-enable */
```

```
const immutableReconciler = (inboundState, originalState, reducedState) => {
const newState = { ...reducedState };

Object.keys(reducedState).forEach(reducerKey => {
if (reducerKey === '_persist' || originalState[reducerKey] !== reducedState[reducerKey]) {
return;
}
newState[reducerKey] = isImmutable(newState[reducerKey])
? newState[reducerKey].merge(inboundState[reducerKey])
: Object.assign(newState[reducerKey], inboundState[reducerKey]);
});

return newState;
};
const immutableTransform = (config = {}) => {
const serializer = Serialize.immutable(Immutable, config.records);
return createTransform(serializer.stringify, serializer.parse, config);
};

const createTransforms = () => Object.keys(entities).map(reducer => createFilter(reducer,
['entities', 'list']));

export const persistConfig = {
key: 'qit/root',
whitelist: ['standards', 'modules', 'lessons'],
transforms: [...createTransforms(), immutableTransform()],
storage,
stateReconciler: immutableReconciler,
version: '1.0.3',
};

const createStore = () => {
const composeEnhancers = composeWithDevTools({
serialize: {
immutable: Immutable,
},
});

const thunkExtraArguments = {
schema,
normalize,
};
```

```
<?php

namespace App\Controller;

use App\Entity\ParentOnlineSession;
use FOS\RestBundle\Controller\FOSRestController;
use Symfony\Component\Form\Extension\Core\Type\SubmitType;
use Symfony\Component\Form\Extension\Core\Type\TextareaType;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\Form\Extension\Core\Type\TextType;
use Symfony\Component\Form\Extension\Core\Type\EmailType;
use Symfony\Component\HttpFoundation\Response;

class AdminController extends FOSRestController
{
public function editLessonContentAction(Request $request, ParentOnlineSession
$parentOnlineSession)
{
$form = $this->createFormBuilder($parentOnlineSession)
->add('content', TextareaType::class)
->add('save',SubmitType::class)
->getForm();

$form->handleRequest($request);
if ($form->isSubmitted() && $form->isValid()) {
$this->getDoctrine()->getManager()->flush();
}

return $this->render('admin/pol_edit_lesson_content.html.twig', [
'form' => $form->createView(),
'index' => $parentOnlineSession->getOrderIndex(),
'emotion' => $parentOnlineSession->getTrajectory()->getType(),
]);
}
public function inviteTherapistAction(Request $request): Response
{
$form = $this->createFormBuilder()
->add('firstName', TextType::class, ['required' => true])
->add('lastName', TextType::class,['required' => true])
->add('email', EmailType::class, ['required' => true])
->getForm();
```

```
$form->handleRequest($request);
if ($form->isSubmitted() && $form->isValid()) {
$email = $form->getData()['email'];
if ($this->get('App\Service\Pub\CheckerService')->checkEmailDuplicate($email)) {
$this->addFlash('error', 'This name is already taken, please choose another one');

return $this->redirectToRoute('admin_invite_therapist');
}


$therapist = $this->get('App\Service\Pub\UserFactoryService')->createTherapist();

$therapist->setEmail($email);
$therapist->setFirstName($form->getData()['firstName']);
$therapist->setLastName($form->getData()['lastName']);
$this->getDoctrine()->getManager()->persist($therapist);
$this->getDoctrine()->getManager()->flush();
$this->get('App\Service\Pub\MailService')->sendTherapistInviteEmail($therapist);
$this->addFlash('notice', "Invitation email was successfully sent to $email"); // TODO move inside service?
//         $this->get('event_dispatcher')->dispatch('parentOnline.newPatient', new NewParentOnlineUserEvent($patient));

return $this->redirectToRoute('homepage');
}

$view = $this->view()
->setTemplate(':admin:admin_invite_therapist.html.twig')
->setTemplateData(['form' => $form->createView()]);

return $this->handleView($view);
}
}
<?php

namespace App\Controller;

use Symfony\Component\HttpFoundation\Request;
use Doctrine\Common\Collections\ArrayCollection;
use FOS\RestBundle\Controller\FOSRestController as Basecontroller;
```